THE VIRTUAL SPACEPLANE:
INTEGRATING MULTIPLE MOTION MODELS
AND HYPERTEXT IN A VIRTUAL ENVIRONMENT

THESIS

Troy D. Johnson
First Lieutenant, USAF

AFIT/GM/ENG/97D-01

19980127 024

THE VIRTUAL SPACEPLANE:
INTEGRATING MULTIPLE MOTION MODELS
AND HYPERTEXT IN A VIRTUAL ENVIRONMENT

THESIS

Presented to the Faculty of the Graduate School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Meteorology

Troy D. Johnson, B.S.

First Lieutenant, USAF

December, 1997

THE VIRTUAL SPACEPLANE:
INTEGRATING MULTIPLE MOTION MODELS
AND HYPERTEXT IN A VIRTUAL ENVIRONMENT

THESIS
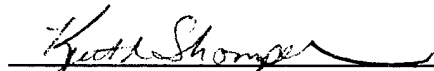
Troy D. Johnson, B.S.

Presented to the Faculty of the Graduate School of Engineering

of the Air Force Institute of Technology
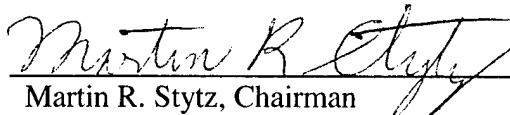
Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Meteorology

_____
Sheila B. Banks
Major, USAF

_____
Keith A. Shomper
Major, USAF

_____
Martin R. Stytz, Chairman
Lieutenant Colonel, USAF

ACKNOWLEDGEMENTS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The Air Force is currently investigating the possibility of developing a manned vehicle capable of operating in space. This Military Spaceplane (MSP) will be capable of ascent to low-earth orbit and maneuvering while in orbit. The goal of this research involved creating the Virtual Spaceplane (VSP), a virtual environment (VE) simulator for the MSP. This thesis examines two ideas significant to virtual environments and cockpit design: multiple motion models and hypertext in a VE.

Movement in a VE has traditionally been modeled using a single motion model. Little work has been done to allow a change of the motion model used during the simulation. This thesis suggests partitioning simulation entities into two sections: the geometry model and the propagation model. This approach is demonstrated in the VSP using multiple propagation models as it transitions from runway to orbit.

This thesis also examines the use of hypertext within a VE. Hypertext has been shown useful for readers to quickly locate information. This thesis will discuss the integration of a hypertext interface into the VSP. The hypertext interface provides checklists, systems status, and consumables status. Hypertext provides the spaceplane pilot with an effective means of referencing large amounts of data.

THE VIRTUAL SPACEPLANE:

INTEGRATING MULTIPLE MOTION MODELS

AND HYPERTEXT IN A VIRTUAL ENVIRONMENT

I – INTRODUCTION

Motivation

Former Chief of Staff of the Air Force Gen. Ronald R. Fogelman, in his roadmap for the future of the Air Force, recently stated that "operations that now focus on air, land and sea will ultimately evolve into space." [FOGE96] He continued by saying, "Air and space forces provide worldwide situation awareness. They are generally the first forces called forward in a crisis." This far-reaching vision will require significant advances in space-related capabilities. At the core of the execution of this vision is a means for cost-effective, responsive and reliable access to and through space [MSIC97].

In response to this goal, the U.S. Air Force has been investigating the possibility of developing a vehicle capable of operating in the space environment. The Military Spaceplane (MSP) is intended to provide a "safe, reliable, operable, supportable, producable, testable, and affordable suborbital and Earth-to-Orbit-and-Return" flight system [VERD97]. The MSP will be capable of accomplishing ascent to low-earth orbit, maneuvering while on orbit, de-orbiting, and performing hypersonic flight within the atmosphere. Takeoff and landing are to be accomplished horizontally from conventional runways.

1

Purpose

The goal of our research is to develop a virtual flight simulator for the Military Spaceplane. The Virtual Spaceplane (VSP) uses virtual environment technologies to simulate many of the capabilities of the proposed MSP. The VSP has the ability to maneuver in the diverse operating regimes from a runway to low-earth orbit. The Virtual Spaceplane also simulates the deployment of a satellite and allows a rendezvous and co-orbit with a space station or low-earth orbiting satellite.

Another research goal during the development of the VSP is to provide the spaceplane with a cockpit interface simpler than found in other advanced air and space vehicles. The cockpits of today's air and spacecraft continue to include more displays and controls. Unfortunately, this increased complexity often interferes with the pilot, rather than providing the intended assistance [HAMM95]. To curb this trend, our research investigates new paradigms of cockpit design, including a hypertext interface used for onboard system diagnostics, checklists, and monitoring consumables.

Thesis Statement

Develop a prototype simulator for the Military Spaceplane capable of horizontal takeoff from a conventional runway, atmospheric flight, low-earth orbit entry, orbital maneuvering, atmospheric reentry, and landing. Investigate new concepts in cockpit design, including virtual environments, reconfigurable displays, and hypertext, while developing the pilot-spaceplane interface.

Scope

The Virtual Spaceplane was developed as a prototype flight simulator. It therefore, implements only a subset of the capabilities of the proposed Military

2

Spaceplane. The VSP simulates the spaceplane's maneuvering in and through the atmosphere and space. It also demonstrates the ability to deploy a satellite into low-earth orbit, and to rendezvous with a satellite or space station.

Also, because the VSP is a prototype for a vehicle that has not yet been developed, no flight characteristics data were available. The VSP, therefore, approximates the flight attributes of a vehicle of the Military Spaceplane's expected size and flight profile. Takeoff and landing are only allowed from a single location.

Some of the displays in the Virtual Spaceplane cockpit were developed only to a limited degree. These prototype displays allowed us to more easily verify the usefulness of various interface techniques. For example, the Engineering Panel's hypertext interface is not implemented using the Hypertext Markup Language (HTML), and the on-line checklists are not intended to be factual, but are representative of the checklists which would be used by the pilot of the Military Spaceplane.

Other limitations include the fact that the rendezvous mission does not include any time restrictions and the satellite deployment mission does not accurately model the boost to a higher orbit typical of many satellite launches. The VSP also is currently implemented with an unlimited fuel capacity. The software was developed to operate on Silicon Graphics Onyx hardware with the Performer execution environment installed.

Approach/Methodology

The Virtual Spaceplane was created using rapid prototyping development techniques. Rapid prototyping allowed us to refine the system requirements during the entire research process. It also allowed the VSP to achieve incremental improvements in

3

the software, and provided us with the ability to determine the effectiveness of the novel cockpit designs used in the spaceplane. Object-oriented techniques are essential to rapid prototyping [STYT97] and were used throughout the implementation of the VSP.

After investigating the architecture used in other large-scale virtual environments, we developed the architecture for the VSP. We also reviewed existing software developed by the AFIT Virtual Environments, Medical Imaging & Computer Graphics Lab. The reuse of portions of the AFIT Virtual Cockpit [ADAM96] and the AFIT Solar System Modeler [WILL96] increased the efficiency of the Virtual Spaceplane development process.

The requirement for the VSP to operate in the atmosphere, low-earth orbit, as well the transition region between air and space necessitated a study of the mechanics in these three dissimilar regions. The software for maneuvering the spaceplane in these three regions was developed from three different sources. Atmospheric flight is modeled using procedures developed for the Virtual Cockpit; routines developed by the Air Force Academy Astronautics Department enabled space flight; and flight through the exo-atmosphere was allowed by incorporating routines developed for the Transatmospheric Vehicle (TAV) project.

Thesis Overview

This thesis is divided into six chapters. Chapter Two provides background in the areas the reader is not assumed to have knowledge, but are necessary to understand this work, including a more detailed description of the Military Spaceplane. Chapter Three identifies the specific requirements upon which our research is based. Chapter Four will

4

discuss the research performed as well as the design and implementation of the software written to fulfill the Virtual Spaceplane's requirements. Chapter Five presents the results of the research. Finally, Chapter Six summarizes our research and recommends areas of future research. Throughout the thesis are references to the *Gryphon*. This nickname is given to the spaceplane vehicle that operates within the VSP environment.

Following the thesis are three appendices that provide supplementary information on implementation details of the VSP. The first two appendices provide details on the implementation of the coordinate conversion utilities developed as part of this research. The third appendix provides a detailed list the progress of the Virtual Spaceplane, including completed work and areas for future research.

II – BACKGROUND

As described in the previous chapter, this thesis will cover the work I accomplished on the Virtual Spaceplane. Before that, however, it is appropriate to briefly discuss some information required to provide a framework this research. First, this thesis will discuss a technology central to the VSP, called virtual reality, followed by a brief discussion of flight simulators. Other work completed at AFIT applicable to our research will then be reviewed. Next, because this research employs many different coordinate systems, I'll present a summary of each system used in the VSP. To provide a background for our vehicle's capability for orbital motion, the following section will provide a brief description of astrodynamics. I'll conclude with a short report of the current space vehicle, the Space Shuttle, followed by a summary of the capabilities of the proposed Military Spaceplane, the basis of our research.

Virtual Reality

Virtual reality (VR) is a relatively new simulation tool that is often used when a conventional, physical mockup is impractical or uneconomical. Virtual reality can be defined as "the creation of the illusion of immersion of the user in a computer-generated environment" [BRYS93]. When in a virtual environment (VE), the user feels surrounded by computer-generated objects moving within a computer-generated environment [STYT97]. The user views the scene through a head-mounted display (HMD) or some other visual display. Some means of tracking head and body movement are usually included to stimulate the users' immersion, or "presence", in the environment.

Research conducted by Pausch, et al, has shown that users of a VE interface perform significantly faster than users of a traditional, fixed monitor and mouse, interface [PAUS97]. Pausch attributes VR's improved performance to the mental frame-of-reference developed because of the presence in the virtual environment.

Virtual reality has proven useful in many areas. Virtual environments have been demonstrated to be an excellent means to explore celestial activity [WILL96]. Probably the most notable source of VR technology has come from work associated with the development of vehicle simulators, especially aircraft flight simulators [MCKI91].

Flight Simulators

The goal of our research was to develop a virtual flight simulator for a prototype air-space vehicle. Flight simulators have been employed since the Wright brothers developed their first airplane. The essential form of flight simulation is the creation of a dynamic representation of the behavior of an aircraft in a manner that allows the human operator to interact with the simulation [ROLF86]. Most computer-based flight simulators use mathematical models of aircraft motion to emulate the aircraft's flight. Flight simulators have been used for aircrew training, new aircraft design, as well as research in advanced avionics. Flight simulators have taken many forms: a physical mockup of the entire aircraft, a physical reproduction of just the cockpit and controls, and mockups that use VE technology. The most costly of all flight simulators are those that provide both motion and visual cues to the pilot, making the simulation as authentic as possible. Virtual flight simulators have been demonstrated to be an inexpensive alternative to these high-end full motion flight simulators [ADAM96].

Previous Relevant AFIT Projects

The AFIT Virtual Environments, Medical Imaging & Computer Graphics Lab has developed a variety of virtual environments. These include the Synthetic BattleBridge [WELL96], the Virtual Emergency Room [GARC96], the Solar System Modeler (previously called the Satellite Modeler and the Space Modeler) [WILL96], and the Virtual Cockpit [ADAM96]. More on these projects can be found at www.afit.af.mil/Schools/EN/graphics/veprojects/ve.html. Two of these, the Solar System Modeler and the Virtual Cockpit, are particularly significant for our research into virtual space flight.

Solar System Modeler

The representation of the environment is an important part of any virtual reality project. Because the space setting is an important part of the VSP, the Solar System Modeler was used as a basis for the VSP's environment. The Solar System Modeler models the location of the Sun, all nine planets, many planetary moons, some of the larger asteroids and comets, plus over 30 thousand stars. The Solar System Modeler also correctly models the full constellation of Global Positioning Satellites as well as interplanetary probes.

Planetary positions are calculated using algorithms described by Meeus [MEEU91], which provide very accurate locations based on the astronomical time. Satellites are propagated using the SGP4 routines developed by the North American Air Defense Command (NORAD), which accurately model the movement of objects in low-earth orbit. The satellites are initialized using two-line element (TLE) sets, which describe the position and movement of the satellite. TLEs are updated routinely and are available via the Internet (a site we found useful is www.grove.net/~tkelso).

Virtual Cockpit

The Virtual Cockpit (VC) provided the basis for atmospheric flight for the Virtual Spaceplane. The VC represents the flight characteristics and cockpits of known, established systems. The VC models the cockpit interior and flight characteristics of either an F-15 or an F-16. The purpose of the VC is to provide pilots with a simulator that can be used for training in the use of aircraft displays and controls. It makes use of a rapidly reconfigurable cockpit, converting the displays as well as flight characteristics between the F-15 and F-16 fighters, even during flight.

The VC uses a set of routines, collectively referred to as the Aero Model to simulate atmospheric flight. The Aero Model was originally implemented in the C programming language for the Wright Laboratory's Flight Simulation Facility, and has since been converted into a C++ class for use in the Virtual Cockpit. The Aero Model is reconfigurable and allows various flight characteristics to be modified. Flight characteristics files for F-15, F-16, F-18, F-5E and A-10 aircraft already exist for this model. Flight characteristics are represented using data files describing the flight responses at various speeds, altitudes, and angles-of-attack.

Coordinate Systems

Many different coordinate systems are used in virtual environments. The Virtual Spaceplane, in particular, uses seven distinct systems in the simulation. Some coordinate systems remain fixed throughout the entire simulation, while others may change in orientation and/or position with time. Some assume a flat earth, while others more accurately model a spherical or ellipsoidal earth.

This section describes each of the coordinate systems used in the simulation. A table summarizing the coordinate systems used in the Virtual Spaceplane is located on page 16.

WGS84 Coordinate System

One round-earth coordinate system is the World Geodetic System 1984 (WGS84). This system, amongst other things, defines an ellipsoid that provides a simplified description of the (actually pear shaped) Earth. The ellipsoid is almost spherical, but is slightly fatter at the equator than near the poles. The WGS84 standard defines the major axis (the radius at the equator) to be 6378.137km, and the polar radius to be 6356.7523km [DOD87][SSA97]. The WGS84 coordinate system is used in many applications, including the Global Positioning System (GPS) as well as distributed virtual environments (see DIS Coordinates below). The WGS84 coordinate system is shown in Figure 1.
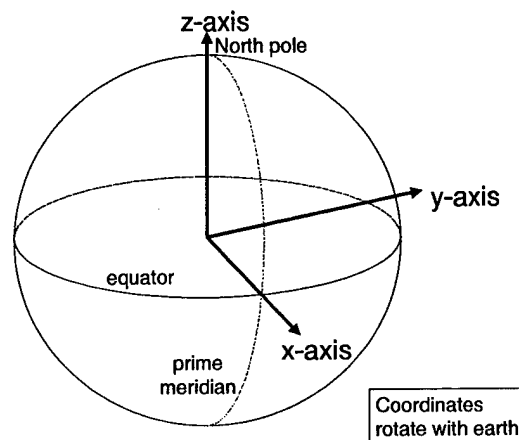
Figure 1 – WGS84 Round Earth Coordinate System

This reference frame is a right-handed coordinate system that has its origin at the center of the Earth [IST93]. The axes are defined such that the z-axis passes through the North Pole. The x and y axes are in the Earth's equatorial plane; the x-axis passes through the prime meridian (0 degrees longitude) and the y-axis passes through 90 degrees east longitude. The WGS84 system is therefore not a fixed, or "inertial", coordinate system because it rotates as the Earth rotates about its axis of rotation. Meters are the standard unit of measurement in the WGS84 coordinate system.

ECI Coordinate System

Another round-earth coordinate system is the Earth Centered Inertial, or ECI coordinate system. As the name implies, the ECI system is an inertial coordinate system whose origin is at the center of the Earth. Similar to the WGS84 system, the x and y axes pass through the equator, and the z-axis passes through the North Pole [BATE71]. However, as shown in Figure 2, the ECI system does not rotate with the Earth. Rather the coordinate system is fixed with respect to the stars and the Earth revolves within the coordinate system. The x-axis is oriented to point in the direction of the vernal equinox. The vernal equinox direction is defined by the vector from the Sun to the Earth on the first day of the northern-hemisphere autumn. Because the ECI system is an inertial coordinate system, it is useful for calculating future positions for objects that move independent from the Earth (e.g., the Sun, Moon, and satellites).
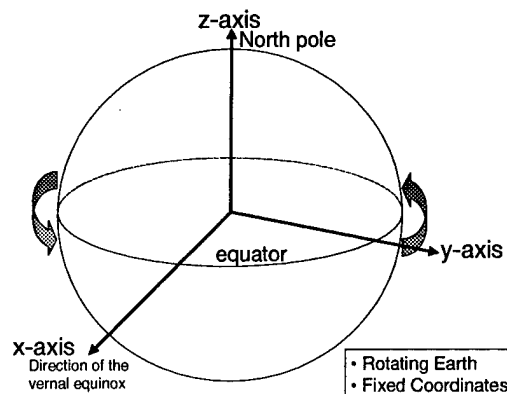
Figure 2 – The ECI Coordinate System

NED Coordinate System

During the simulation, it is often convenient to represent the position and orientation of an object over a flat plane. The North-East-Down (NED) coordinate system is one such fixed, flat-earth coordinate system. The location of the origin in the NED coordinate system is arbitrary. The NED system is a right-handed coordinate system in which the x-axis points north, the y-axis points

east, and the z-axis points down (see Figure 3). The NED system is the primary coordinate system

used in the Virtual Cockpit.
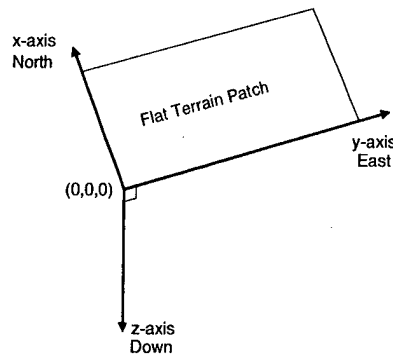
Figure 3 – NED Coordinate System

ENV Coordinate System

Another flat-earth coordinate system used in some virtual environments is the East-North-

Vertical (ENV) system. This coordinate system is similar to the NED system in that it is also a

fixed right-handed coordinate system, but has the x-axis oriented east, the y-axis oriented north, and

the z-axis oriented up (see Figure 4). Although the origin of the ENV system is arbitrary, the VSP

chooses the origin to be at the surface of the Earth at 0 degrees latitude, and 0 degrees longitude.
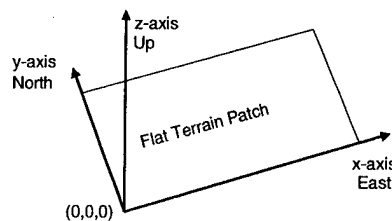
Figure 4 – ENV Coordinate System

Geodetic Coordinates

An object's position may also be designated by specifying the latitude, longitude and altitude [TOMS93]. The Geodetic origin is located at sea level at the intersection of the Prime Meridian and the equator. The units of measurement for the geodetic coordinate system are commonly degrees (for latitude and longitude) and meters (for altitude). Lines of latitude range from −90 to +90 (positive values north of the equator). Lines of longitude range from −180 to +180 (positive values east of the prime meridian). Altitude is positive above the surface of the Earth. When projected rectilinearly onto two dimensions, the familiar Mercator map projection is obtained (see Figure 5).
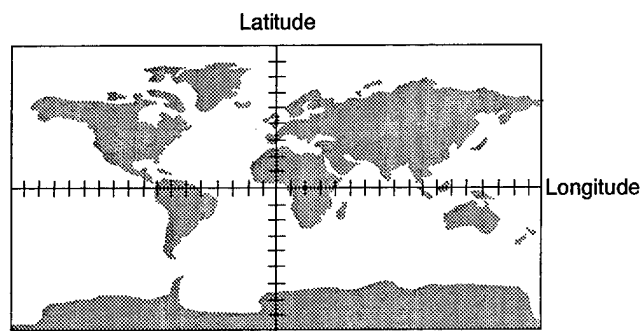
Figure 5 – Geodetic Coordinate System

Performer Object Coordinates

Performer is a graphics programming environment used with Silicon Graphics™ hardware. In addition to providing a complete library of tools for drawing objects in a three-dimensional scene, it also furnishes a strong library of vector and matrix math utilities.

Objects in Performer are modeled with the origin at the object's center of geometry. The positive y-axis is "forward", the z-axis is "up", and the x-axis points to the "right" [SGI95]. This

13

right-handed coordinate system moves with the object. Performer uses heading, pitch and roll to represent the object's orientation in this coordinate system:

Heading   specifies rotation about the z-axis.

Pitch   specifies rotation about the x-axis.

Roll   specifies rotation about the y-axis.

The object is first rotated about the z-axis, followed by rotation about the x-axis, and finally by rotation about the y-axis. Figure 6 shows the Performer object coordinate system.
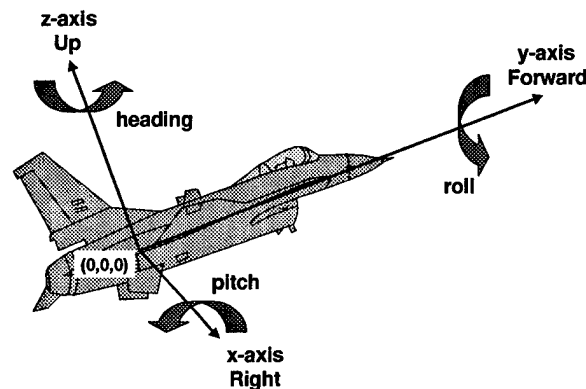
Figure 6 – Performer Object Coordinate System

DIS Coordinates

The DIS standard requires that positional state information be transmitted using a coordinate system identical to the WGS84 coordinate system [LOPE93]. An object's orientation is defined differently than the system used by Performer (see Figure 7). Using the DIS standard, an object is first rotated about the z-axis, followed by rotation about the y-axis, and finally by rotation about the x-axis. The angles designating the amount of rotation about the various axes are sometimes referred to as Euler angles:

Psi (ψ)   specifies rotation about the z-axis.

Theta (θ)   specifies rotation about the y-axis.

Phi (φ)   specifies rotation about the x-axis.
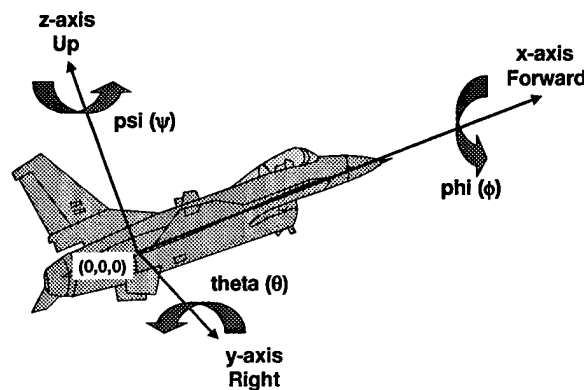
Figure 7 – DIS Orientation Coordinates

All of the coordinate systems described above are used in the Virtual Spaceplane. The WGS84 system is the primary coordinate system used in the VSP. The VSP must have the capability to convert between the many coordinate systems to use routines designed for the different systems. Table 1 summarizes the coordinate systems described above.

Table 1 – Summary of Coordinate Systems used in VSP

Coordinate System	Description	Use in VSP
WGS84	Round-earth system that rotates with Earth.	Primary coordinate system in VSP.
ECI (Earth Centered Inertial)	Round-earth system fixed with respect to the stars.	Used for movement in space.
NED (North-East-Down)	Flat-earth system used in Virtual Cockpit.	The Aero Model's coordinate system. Used for movement in atmosphere.
ENV (East-North-Vertical)	Flat-earth system.	Intermediate system use in conversion from NED to WGS84.
Geodetic	Latitude, Longitude and Altitude.	Used in coordinate conversion and for navigation.
Performer Object	Specifies the orientation of an object (heading, pitch, and roll)	Drawing objects in the scene.
DIS (Distributed Interactive Simulation)	Position is same as WGS84. Orientation is specified using Euler angles (ψ, θ, ϕ).	Allows participation in a distributed simulation.

Fundamentals of Astrodynamics

A majority of the Spaceplane's mission time will be in orbit around the Earth. A short summary of the science of bodies in space, or astrodynamics, will therefore be presented in this section. Johann Kepler is often considered the father of astrodynamics. In the early 17th century, Kepler discovered three laws that govern planetary motion [BATE71]. Kepler's laws are the following:

First Law: The orbit of each planet is an ellipse, with the Sun at a focus.

Second Law: The line joining the planet to the Sun sweeps out equal areas in equal times.

Third Law: The square of the period of a planet is proportional to the cube of its mean distance from the Sun.

The motion of a satellite around the Earth (or a planet around the Sun) is confined to a plane fixed in space, referred to as the orbital plane. The satellite's position is described using the polar

16

equation for an ellipse (see Figure 8). The vector, **r**, specifies the position of the satellite with respect to the Earth (the Earth is located at the focus, F1). The ellipse's second focus, F2, is unoccupied. The polar angle between **r** and the point on the conic nearest the focus is represented by v. In the equation relating **r** and v, p is a geometrical constant of the conical section called the "parameter", or "semi-latus rectum". The constant e is called the eccentricity, and it determines the shape of the conic section.

$$r = \frac{p}{1 + e\cos v}$$

e = 0	Circle
0 < e < 1	Ellipse
e = 1	Parabola
e > 1	Hyperbola
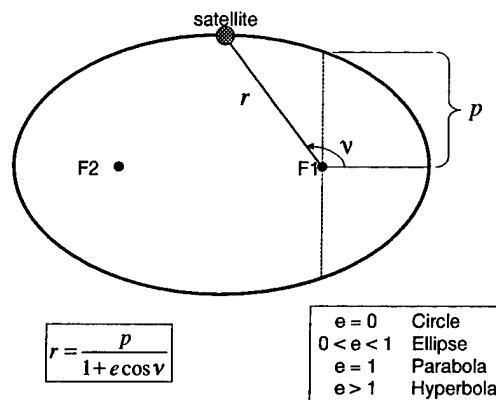
Figure 8 – Geometry of an ellipse

These principles, and other laws of astrodynamics, are used in the VSP to move satellites, as well as the spaceplane itself, in orbit around the Earth. For an excellent introduction into astrodynamics, refer to Bate, et al [BATE71].

Orbital elements

The orbit of a satellite around the Earth may be completely defined by six independent parameters, referred to as the classical orbital elements (see Figure 9). Table 2 provides a short description of each of the most commonly used elements. The orbital elements are based on the ECI coordinate system described above.

Table 2 – Orbital Elements

Name of Orbital Element	Symbol	Description
Semi-major axis	a	Defines the size of the orbit
Eccentricity	e	Defines the shape of the orbit
Inclination	i	The angle between the orbital plane and the equatorial plane
Longitude of ascending node	Ω	The angle between the vernal equinox direction and the line of nodes (the intersection between the equatorial and orbital plane)
Argument of periapsis	ω	The angle between the line of nodes and the periapsis point
Time of periapsis passage	T	The time when the satellite was at periapsis
True Anomaly *	ν	The angle from periapsis to the satellite's current position
Argument of latitude *	u	$u = \omega + \nu$
Longitude of periapsis *	Π	$\Pi = \Omega + \omega$
True longitude *	ℓ	$\ell = \Omega + u$
* – Secondary orbital element (dependent on the first six)		

The true anomaly, ν, is often used in place of T as one of the six primary orbital elements [BATE71]. The Solar System Modeler (and the Virtual Spaceplane) use this convention for calculation of positions for the planets. For objects orbiting the Earth, it is customary to replace the term periapsis with perigee.

18
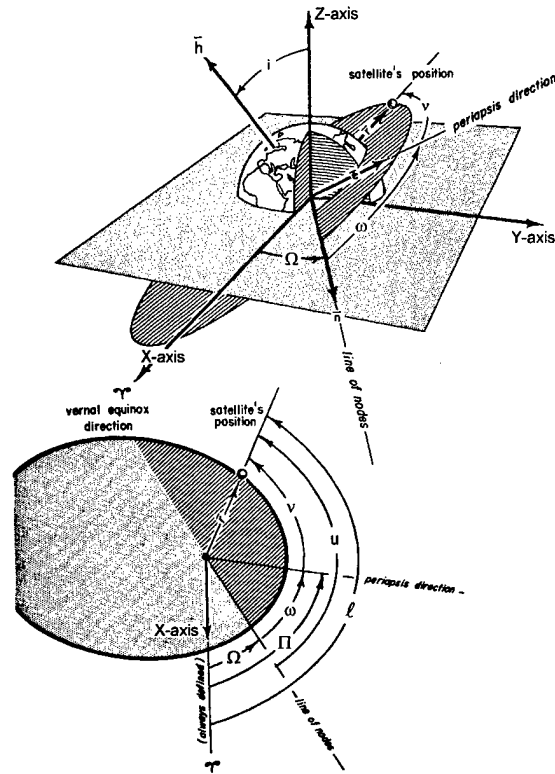
Figure 9 – Orbital elements [from Bate 71]

Two-Line Elements

NORAD maintains general perturbation element sets on all resident space objects. These element sets are compiled into two lines of text data for distribution. These Two-Line Element (TLE) sets consist of 24 fields of information, some of which are similar to the classical orbital elements described above. Converting from TLE data to the classical orbital elements is fairly straightforward, but obtaining the additional fields of information specified by a TLE from the six classical orbital elements is more difficult. It is also important to note that the TLE sets are "mean" values obtained by removing periodic variations. In order to obtain the most accurate predictions, these periodic variations must be reconstructed (by the prediction model) in exactly the same way

they were removed by NORAD. Hence, inputting TLE sets into a different model will result in degraded predictions [HOOT80].

NASA Space Shuttle

Before investigating a new model for space flight, it is appropriate to review the methods used today. The primary manned vehicle currently used for low-earth orbit is the Space Shuttle. Design of the Space Shuttle began in the early 1970s. It is the first reusable launch vehicle in the United States space program.

The Shuttle cockpit is one of the most complex cockpits ever assembled. The flight deck has 65 panels containing over 1700 buttons, knobs, switches, gauges and display screens [NASA94]. On most flights, two astronauts (a pilot and a flight commander) operate the orbiter. Due to lack of cockpit space, many of the controls and much of the information in the cockpit are behind, and often out of reach of the pilot and flight commander. In addition, because of the complexity of the interface, two or more astronauts are required to execute some shuttle maneuvers [KERR88].

The Space Shuttle was designed for a vertical takeoff using two solid-fuel booster rockets, plus an external liquid-fuel tank. After liftoff, the mission control center at the Johnson Space Center in Houston assists the space shuttle with orbit and reentry maneuvering, and is also responsible for monitoring the spacecraft's systems during the flight. After the mission has been completed, the orbiter enters the Earth's atmosphere and lands on a runway like an airplane.

Military Spaceplane

The goal of the research for this thesis was to develop a flight simulator for the Military Spaceplane, an air-space vehicle being developed by the United States Air Force. General

Fogelman, in his roadmap for the future of the Air Force, stated that "operations that now focus on air, land and sea will ultimately evolve into space." [FOGE96] The U.S. Air Force has recently been investigating the possibility of developing a vehicle capable of operating in and through the space environment. Interest in a Military Spaceplane (MSP) dates from initial research into reusable launch vehicles (RLV) such as the X-20/Dyna-Soar program in the late 1950's, to more recently, the National Aerospace Plane (NASP) and the Delta-Clipper (DC-X) in the early 1990's [USC90]. The MSP is envisioned to be capable of sustaining an operational tempo like a military aircraft. The MSP must therefore have a rapid turn-around time and be easily maintainable [VERD97]. One means of providing such a rapid turn-around time is single-stage-to-orbit (SSTO) technology. A SSTO vehicle does not use external tanks or rockets, and takes off with all the fuel needed to achieve orbit, accomplish its mission and return to Earth. Sortie turnaround processing is therefore limited to the more rapid tasks of refueling, loading new mission assets, and systems checkout [KUPE92].

The MSP will be capable of accomplishing ascent to low-earth orbit, maneuvering while in orbit, de-orbiting, and performing hypersonic flight within the atmosphere. Takeoff and landing are accomplished horizontally from conventional runways. The vehicle should be capable of manned, unmanned virtually-piloted, as well as completely autonomous flight. Current projections call for initial production of a military spaceplane beginning around 2010 [MSIC97].

Summary

This chapter provided a summary of previous work and a synopsis of information that should make the remaining portions of this thesis easier to read. The next chapter will describe the requirements of the Virtual Spaceplane.

III – REQUIREMENTS

Before proceeding into the Design and Implementation chapter, I'll discuss more fully the requirements of our research project. These requirements derive from the thesis statement given in Chapter One and evolved during the course of the development of the VSP.

Our goal is to develop a simulator for an aerospace vehicle capable of simulating operation in the atmosphere as well as in low-earth orbit. This simulator must be capable of accurately simulating the flight characteristics of a vehicle in both flight regimes, as well as the tenuous exo-atmospheric region. It must also model takeoffs and landings. The VSP should provide a convincing environment in which to maneuver. Finally, we are developing a framework for a future spaceplane user interface.

The VSP requirements are divided into five primary areas: simulated capabilities of the MSP, supported missions, user interface, virtual environment, and miscellaneous requirements. The following sections briefly describe each area. Because this research was conducted in cooperation with two other students, only a portion of these requirements will be addressed in the following chapters. Readers interested in the remaining requirements are encouraged to investigate the companion theses by Capt John Lewis and Lt Scott Rothermel [LEWI97][ROTH97]. A more detailed requirement list is given in Appendix C.

Simulated Capabilities of MSP

The capabilities to simulate the flight of the Military Spaceplane from horizontal takeoff to low-earth orbit play a significant role in the development of the Virtual Spaceplane. Table 3 lists the capability sub-tasks required of the VSP.

Table 3 – Capability Requirements.

ID	Requirement Description
	Flight Characteristics
1.11	The spaceplane shall simulate maneuvering on runways.
1.12	The spaceplane shall simulate flight through the atmosphere.
1.13	The spaceplane shall simulate maneuvering in space.
1.14	The spaceplane shall transition from one flight regime to another when appropriate.
	Manual Operation
1.21	The VSP shall provide capability to manually operate in the atmosphere.
1.22	The VSP shall provide capability to manually operate in space.
	Automatic Operation
1.31	The VSP shall provide capability to automatically takeoff.
1.32	The VSP shall provide capability to automatically fly specified routes.
1.33	The VSP shall provide capability to automatically enter orbit.
1.34	The VSP shall provide capability to automatically modify orbital parameters.
1.35	The VSP shall provide capability to automatically reenter the atmosphere.
1.36	The VSP shall provide capability to automatically land.

Thesis focus

In this thesis, I will discuss the fulfillment of all of these requirements. Methods for manually and automatically maneuvering the VSP in all regions of operation will be discussed in the next chapter. Rothermel discusses the architecture on which this will be built [ROTH97]. Details on the interface to these procedures are covered by Lewis [LEWI97].

Supported Missions

The Military Spaceplane is intended to facilitate many types of missions for the exploitation of space to achieve military objectives. The VSP initially only supports a subset of these mission profiles as listed in Table 4. Future work should expand the VSP mission capabilities significantly.

Table 4 – Mission Requirements.

ID	Requirement Description
	Supported Missions
2.1	The VSP shall support rendezvous with orbiting objects.
2.2	The VSP shall support deployment of satellite.

Thesis focus

In the following chapters, I will discuss both requirements 2.1 and 2.2. A detailed description of the interface used to execute a rendezvous and deployment are discussed by Lewis [LEWI97].

User Interface

A primary goal of the VSP was to investigate unconventional interface schemes for controlling a MSP and for maintaining the pilot's situational awareness without overburdening the pilot with tasks. The requirements specifying the interface goals are listed in Table 5.

Table 5 – User Interface Requirements.

ID	Requirement Description
	Interaction methods
3.11	All interface functionality shall be available via a three-button mouse.
3.12	The VSP shall support a head-mounted display (HMD) with head tracking.
3.13	Auxiliary functionality will be available via the keyboard.
	Configurable Cockpit
3.21	The user shall be able to selectively display information interactively.
3.22	The user shall be able to modify the location of information displays interactively.
	Displayed Information
3.31	The interface shall display state information of the Gryphon in the atmosphere.
3.32	The interface shall display state information of the Gryphon during orbit entry/reentry.
3.33	The interface shall display state information of the Gryphon in space.
3.34	The interface shall display status of consumables (propellants, life-support, etc.).
3.35	The interface shall display state information of potential targets.
3.36	The interface shall assist the user in locating/acquiring potential targets.
3.37	The interface shall assist the user with system management and diagnosis.
3.38	The interface shall investigate hypertext paradigms for display of information.
3.39	The interface shall minimize obstruction of the user's view.
	Controlling the Gryphon
3.41	The interface shall not utilize a throttle and stick for control of Gryphon.
3.42	The interface shall enable users to change the state of the Gryphon in the atmosphere.
3.43	The interface shall enable users to change the state of the Gryphon in space.

Thesis focus

The investigation of hypertext paradigms for the display of information, requirement 3.38, will be covered in the following chapters. The hypertext interface will facilitate the display of consumables status, requirement 3.34. Lewis covers the remaining interface requirements [LEWI97].

25

Virtual Environment

An effective virtual environment must encourage the user to make the mental transition from participating in a computer simulation to accepting the perception that they are within the portrayed environment. The Virtual Environment requirements in Table 6 are intended to enhance the sense of presence for user of the VSP.

Table 6 – Environmental Requirements.

ID	Requirement Description
	Environment
4.1	The VSP shall present convincing terrain surrounding the landing site at Edwards AFB.
4.2	The VSP shall present convincing representations of the Earth, Sun, and Moon.
4.3	The VSP shall simulate multiple constellations of Earth orbiting objects.
4.4	The VSP shall portray the transition between day/night and atmosphere/space.

Thesis focus

This thesis will specifically address completion of 4.2 and 4.3, involving producing both the movement characteristics and visual appearance of the Earth, Sun, Moon and satellites.

Miscellaneous

Several requirements for the VSP did not correspond to one of the previous areas. These requirements are listed in Table 7.

Table 7 – Miscellaneous Requirements.

ID	Requirement Description
	Miscellaneous
5.1	The VSP shall accept state information of remote entities via the DIS protocols.
5.2	The VSP shall transmit state information of the Gryphon via the DIS protocols.
5.3	The VSP shall operate at a mean of 15 frames per second on a 4 processor 250 MHz R10000 SGI Onyx2 with Infinite Reality graphics equipped with 16 Mbytes of hardware texture memory.

Thesis focus

This thesis will not discuss these three requirements. Rothermel provides a discussion of the miscellaneous requirements [ROTH97].

Summary

In this chapter, I discussed the requirements of the Virtual Spaceplane project. We divided the requirements into five areas: simulated capabilities of the Military Spaceplane, the missions supported by the VSP, plus user interface, virtual environment, and other miscellaneous requirements. The next chapter will cover the research we performed to accomplish the research goals.

IV – DESIGN & IMPLEMENTATION

The previous chapter covered the requirements of the Virtual Spaceplane project and alluded to those that this thesis will address. This chapter will cover the research done during the design and implementation of the Virtual Spaceplane. First, we'll discuss the software architecture, and the division of simulation entities into a geometry model and a propagation model. The topic of propagation models, the mathematical models used to move simulation entities through the virtual environment, will then be fully explored. Following a description of each propagation model used in the VSP is a summary of the techniques used by the spaceplane to rendezvous with a satellite in orbit. The next section describes the implementation of the spaceplane with the capacity to transition between the various propagation models as well as operating the landing gear and payload doors. Next, the thesis will describe a set of utilities developed to allow conversion among the many coordinate systems used in the VSP. The implementation of another portion of the VSP, the autopilot, is then presented. In the final section, we'll discuss an implementation of a hypertext interface in the VSP virtual environment.

The Software Architecture

An important part of any program design is the software infrastructure. Before getting to the details of my research, I'll provide a short summary of the Virtual Spaceplane's architecture to provide a framework for the rest of the chapter. Rothermel provides a more detailed examination of the VSP's architecture [ROTH97].

The main loop of the simulation is the GO method inside SIM (see Figure 10). The RENDERER is responsible for drawing the virtual environment. User input is handled through

various IO_MODIFIERS. Each input device updates the Common Object Database (CODB) with

user inputs. The SIM then reads the CODB during its `TranslateInputs` method. When SIM is

initialized, it instantiates the COCKPIT and other SIMOBJECTS. Each SIMOBJECT has an associated

PROPMODEL that allows the SIMOBJECT to move in the simulation. The relationship between a

SIMOBJECT and a PROPMODEL will be discussed in the next section. The SIMCLOCK tracks the

passage of simulation time and allows the rate at which time passes to be adjusted. A SIMOBJECT

(in particular, the spaceplane) can also be controlled by the AUTOPILOT. When flying in the

atmosphere, the AUTOPILOT follows a ROUTE of WAYPOINTS as it moves toward its destination.

The AUTOPILOT will be covered in more detail later in this chapter. The COCKPIT groups related

information into panels that can be moved as the pilot desires. The hypertext panel was developed

to investigate the use of hypertext techniques in a virtual environment. This thesis will report on

our research of these hypertext ideas beginning on page 67.
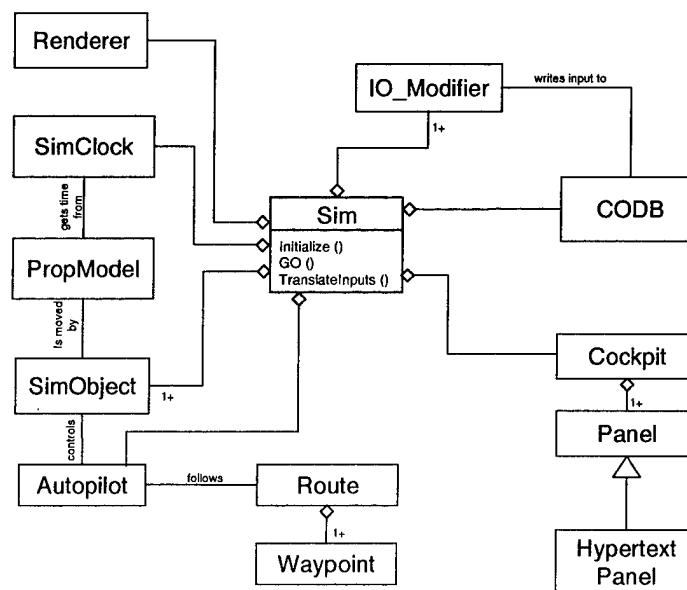
Figure 10 – Software Architecture of the Virtual Spaceplane

Relationship between a SimObject and PropModel

The key to the implementation of the various means of moving the spaceplane through the air and space environments is the relationship between a simulation object and a propagation model. Each entity's geometry is modeled using the SIMOBJECT class, whereas its movement is modeled using the PROPMODEL class. As Figure 11 illustrates, both the SIMOBJECT and PROPMODEL classes have many sub-classes. Each SIMOBJECT is moved by its corresponding PROPMODEL. All propagation models get the simulation time from the simulation timer (SIMCLOCK). The spaceplane uses multiple propagation models to model its flight during the simulation. Therefore, the spaceplane, also referred to as the Gryphon, requires the capability to transition between its various propagation models to accomplish flight from the Earth's surface to Earth orbit. The Gryphon, and the transition between propagation models, will be discussed in more detail later in this chapter.
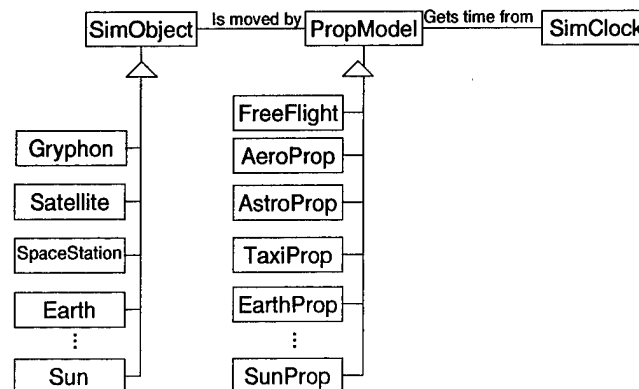
Figure 11 – Relationship between SIMOBJECT and PROPMODEL

30

Propagation Models

Separating the functionality for a simulation entity's motion representation from its geometry representation in the Virtual Spaceplane allows the Gryphon to change propagation models during the simulation. This capability permitted the spaceplane to maneuver in the disparate environments of the runway, atmosphere, and space.

As shown in Figure 11, each simulation object has an associated propagation model responsible for modeling the movement of the entity during the simulation. Once per frame, each SIMOBJECT directs its corresponding PROPMODEL to compute the entity's position based on the amount of simulation time that has passed since the previous update.

The PROPMODEL's `Initialize` method (see Figure 12) is used to set the initial position, orientation and speed of the associated SIMOBJECT. Once per frame, the PROPMODEL's `CalcNewPosition` method is called, providing the propagation model the opportunity to determine a new position based on its speed and previous position. The remaining PROPMODEL methods are used to access the position and orientation information computed during `CalcNewPosition`. `GetPosition`, `GetOrientation`, and `GetSpeed` return the current position, orientation, and speed respectively. The three-dimensional velocity can be obtained from the `GetVelocity` method. `GetInstrumentOri` allows the caller to get the orientation in the ENV coordinate system. The instrument orientation is the heading of the object with respect to true north, and is used in many of the cockpit displays. As the name implies, the `GetGroundSpeed` method returns the ground speed, also used primarily for cockpit displays. `GetMach` returns the speed with respect to the speed of sound. Although the `GetMach` procedure written for the PROPMODEL class assumes a fixed speed of sound (343.0 m/s), a subclass of PROPMODEL (for example AEROPROP) can override this assumption and compensate for the fact that the mach

31

number depends on the atmospheric density. The last method, `GetPropModelType`, allows the simulation to determine which propagation model a SIMOBJECT is currently using. Both the AUTOPILOT (page 62) and the SIMGRYPHON (page 49) use `GetPropModelType` to modify their behavior depending on the propagation model currently being used by the spaceplane.

PropModel
Initialize ()
CalcNewPosition ()
GetPosition ()
GetOrientation ()
GetSpeed ()
GetVelocity ()
GetInstrumentOri ()
GetGroundSpeed ()
GetMach ()
GetAltitude ()
GetPropModelType ()

Figure 12 – Methods common to all propagation models

Many different PROPMODELS were required by the VSP due to the diverse regimes in which simulation entities operate: FREEFLIGHT, which ignores many laws of motion, AEROPROP for atmospheric flight, ASTROPROP for Earth orbit, TAXIPROP for runway operations, as well as propagation models for the Sun, Earth, and Moon.

FreeFlight

The first propagation model developed, FREEFLIGHT, ignores many of the laws of motion such as the concepts of mass, acceleration, and inertia. Maneuvering is performed by making calls to the `TurnLeft`/`TurnRight`, `PitchUp`/`PitchDown`, and `RollLeft`/`RollRight` methods and by setting the speed (see Figure 13). All orientation and speed changes are performed instantaneously.

32

```
┌─────────────────────┐
│     FreeFlight      │
├─────────────────────┤
│ SpeedUp ()          │
│ SpeedDown ()        │
│ SetSpeed ()         │
│ Stop ()             │
│ PitchUp ()          │
│ PitchDown ()        │
│ RollRight ()        │
│ RollLeft ()         │
│ TurnRight ()        │
│ TurnLeft ()         │
└─────────────────────┘
```

Figure 13 – Methods in the FREEFLIGHT propagation model

The FREEFLIGHT propagation model was initially developed for testing of the VSP. Another use of FREEFLIGHT arose because of the architecture design specification that all SIMOBJECTS must have an associated propagation model. Therefore, objects such as the terrain and stars that do not move during the simulation use FREEFLIGHT as a placeholder model.

AeroProp

Although atmospheric flight is not a primary concern for the Spaceplane, we must still provide the capability to fly the VSP using a standard aircraft dynamics model. This propagation model, called AEROPROP, is based on the Aero Model, currently used in the AFIT Virtual Cockpit [ADAM96].

Figure 14 shows the methods provided by AEROPROP which allow the VSP to set the various control surfaces of the spaceplane. The throttle is controlled using the ThrottleUp, ThrottleDown, and SetThrottle methods. The aircraft's control surfaces are adjusted using the remaining methods. The pitch, rudder and bank all vary between –1.0 and +1.0. The throttle varies from 0.0 (off) to 1.5 (full throttle), with 1.0 representing full military power.

```
┌─────────────────────┐
│     AeroProp        │
├─────────────────────┤
│ ThrottleUp ()       │
│ ThrottleDown ()     │
│ SetThrottle ()      │
│ SetPitch ()         │
│ PitchUp ()          │
│ PitchDown ()        │
│ SetRudder ()        │
│ RudderLeft ()       │
│ RudderRight ()      │
│ SetBank ()          │
│ BankLeft ()         │
│ BankRight ()        │
└─────────────────────┘
```

Figure 14 – Methods in the AEROPROP propagation model

AstroProp

The VSP must accurately model low-earth orbits with trajectories similar to those routinely executed by the Space Shuttle. Several techniques exist for forecasting the position of an object in orbit. Potential methods investigated for use in the VSP included the SGP4 algorithm and the use of classical orbital mechanics.

SGP4

The first method we examined was based on a mathematical model developed by Ken Cranford in 1970 for predicting the future position and velocity of satellites [LANE79][HOOT80]. This routine, called SGP4, takes into account the irregular shape of the Earth, as well as atmospheric drag on the satellite.

This approach has several advantages. The SGP4 algorithm very accurately models small perturbations caused by the interactions between the object in orbit and the earth-atmosphere system. A second advantage is that the SGP4 model is initialized using a two-line element (TLE) set. TLEs are available for many objects currently in space. Also to our advantage, the algorithm is already implemented in the AFIT Solar System Modeler for the propagation of satellites.

We decided against using this method for two reasons. First, the algorithm can't be used to compute how the orbit changes when a maneuvering thrust (delta-v) is applied. Second, this method can *only* be initialized using a two-line element set. Because no TLE set exists for the spaceplane as it enters orbit, we would have no capability to initialize the spaceplane using the SGP4 model.

Classical Orbital Mechanics

Another means of determining the future position of an object in orbit is based on the classical orbital elements. Chapter two described how orbital elements might be used to determine the position of a satellite. A satellite being propagated using the classical orbital elements may also be initialized using a TLE. This approach is very convenient, as TLEs are available via the Internet for all the satellites we intend to model in the VSP.

A primary advantage of using orbital elements for space propagation is the ease of initializing the model during orbit entry. Bate, et al, describes a method for determining the six orbital elements based on position and velocity (needed for orbital entry) [BATE71]. This method allows the VSP to transition from the atmospheric regime to space.

Another advantage of the orbital element method is the ability to model the application of maneuvering thrust. When a space vehicle changes its orbit by firing its thrusters, the maneuver is often called a delta-v burn. The thruster, in effect, provides a change to the orbiter's current velocity. As a first approximation, a delta-v burn is often treated as an impulse change in velocity (the thrust is applied instantaneously). Using an impulsive delta-v is a reasonable approximation, used even for a Space Shuttle in orbit, because the thruster burn time is very small compared to the time between burns, and provides accurate results [BLOO74]. By using these approximations, the

problem of applying a maneuvering thrust is reduced to adding the desired delta-v to the current

velocity vector to obtain the new velocity (see Figure 15).

```
1) Compute current position and velocity based
   on orbital elements.
2) Add delta-v to current velocity vector to
   obtain new velocity.
3) Determine the new orbital elements based on
   the new position and velocity.
```

Figure 15 – Algorithm for applying a maneuvering thrust while in orbit

The VSP also uses some of the orbital elements such as the semi-major axis and eccentricity

to quickly determine the shape of the orbit. This technique can be used to graphically represent the

orbit of objects for some of the VSP displays.

Design Choice: Orbital Elements

While the SGP4 method provides a more accurate method of determining the position of a

satellite than the orbital elements based approximation, it provides no ability to modify the

parameters. The ability to modify orbital parameters is an important requirement of the VSP, as it

allows the VSP to maneuver while in orbit. The VSP therefore uses classical orbital mechanics for

its propagation model in space. This propagation model, called ASTROPROP, is based on a library

of FORTRAN routines provided by the US Air Force Academy Department of Astronautics,

written by Capt Dave Vallado. Included in the FORTRAN library are subroutines to convert from

orbital elements to position-and-velocity (and back), subroutines for propagating a satellite, and for

determining the correct delta-v maneuvers to accomplish a rendezvous with another satellite

(described below). Several of the subroutines used in ASTROPROP use an iterative approach to

solve for orbital parameters as well as for delta-v's. These iterative solutions seem to converge faster, and work best at altitudes of 50km or higher. These routines may work best at higher altitudes because they were designed to model motion of objects in low-earth orbit or *higher*.

Figure 16 identifies the methods used by AstroProp to maneuver the spaceplane while it is in orbit. The three throttle functions (`ThrottleUp`, `ThrottleDown` and `SetThrottle`) can be used to provide constant thrust to the Spaceplane. However, whereas constant thrust works well in the atmosphere where pilots have developed an intuitive idea of the motion of the aircraft, an orbiting spaceplane does not move in such an intuitive manner. The VSP therefore does not use this functionality, and implements all thrusts as instantaneous changes in velocity using `ApplyDeltaV`. The remaining methods are used to turn, pitch and roll the associated SimObject. The AstroProp propagation model is also used to model the movement of satellites around the Earth.

AstroProp
ThrottleUp ()
ThrottleDown ()
SetThrottle ()
ApplyDeltaV ()
TurnLeft ()
TurnRight ()
PitchUp ()
PitchDown ()
RollRight ()
RollLeft ()

Figure 16 – Methods in the AstroProp propagation model

OrbitEntryProp

Ideally, the Gryphon would use a single propagation model in the atmosphere as well as in space. This approach would provide seamless operation throughout the entire simulation environment and obviate the need to transition between models. However, because the VSP is only

a prototype, no single propagation model has been developed that is capable of modeling its flight from takeoff, through the atmosphere, to low-earth orbit and back. The transition between the atmospheric flight and orbital maneuvering presented a significant problem during the development of the VSP. The Aero Model used in the atmosphere fails around 30 kilometers. At such high altitudes, the air needed to drive the simulated jet engines of the model is too thin to efficiently burn the fuel. Therefore, a transition to another propagation model must be performed at, or below, this altitude.

An initial attempt at the air-to-space transition involved performing a direct changeover from AEROPROP to ASTROPROP at an altitude of 30km. This approach proved unsatisfactory for two reasons. First, as described above, the iterative techniques used in ASTROPROP model do not work well at altitudes this low. Second, and perhaps as a consequence of the first problem, the transition was quite noticeable and failed to achieve our goal of a seamless flight from runway to low-earth orbit. As Figure 17 shows, each propagation model is accurate only over a portion of the spaceplane's operation regions.
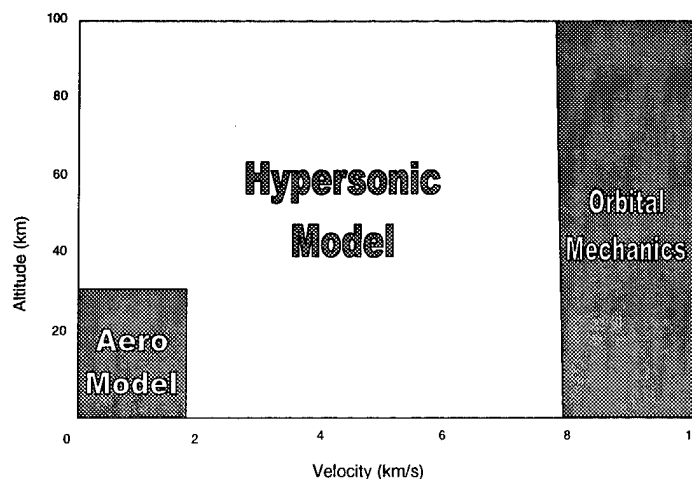
Figure 17 – Each propagation model is accurate only in specific regions

An alternative approach of providing an intermediate propagation model grew out of the difficulties with the direct transition method. The new transition model, called ORBITENTRYPROP, is designed to be fast, though not necessarily completely accurate. This model is also referred to as the exo-atmospheric model, or the hypersonic model, because of its scope. ORBITENTRYPROP is based on a FORTRAN program called QuikFlight written by Mark Goodman and John Livingston from the Aeronautical Systems Center (ASC/XRD) as part of the TransAtmospheric Vehicle (TAV) studies.

While the ORBITENTRYPROP model is active, the spaceplane has no ability to maneuver. The position of the Gryphon is computed using the `GetDownrange` and `GetAltitude` methods (see Figure 18), which returns the horizontal distance the spaceplane has traveled. ORBITENTRYPROP provides two other methods, `GetAngleOfAttack` and `GetWeight`, that can be used by the cockpit to update displays.

OrbitEntryProp
GetDownrange ()
GetAltitude ()
GetAngleOfAttack ()
GetWeight ()

Figure 18 – Methods in the ORBITENTRYPROP propagation model

The routine accepts several tables for input. The tables specify aircraft parameters and the projected flight path. The first table specifies the aircraft's aerodynamic properties, including the coefficients of lift and drag with respect to the angle of attack.

Although the proposed method for the Military Spaceplane's flight through the exo-atmospheric region is using scramjet technology, we had no model to represent this type of engine. Therefore, flight through the upper atmosphere is simulated using rockets. The rockets' specifications are also table-based and include fuel flow, maximum thrust in a vacuum, and the

rocket's exit area. The ORBITENTRYPROP model models flight using two large rockets that provide most of the thrust, and three smaller rockets (similar to the configuration of the Space Shuttle). The rocket model takes into account altitude when calculating thrust (rockets produce less thrust at low altitudes/high atmospheric pressures) and drag (additional drag is produced when the rockets are off).

The last table specifies the planned trajectory, given as a set of points in the altitude vs. velocity domain. Figure 19 shows a trajectory through the upper atmosphere. Velocity in kilometers-per-second is displayed on the ordinate, and altitude in kilometers is the abscissa. The thick line represents the spaceplane's trajectory. At lower altitudes, the spaceplane maintains constant dynamic pressure. Dynamic pressure is related to both altitude (because of the atmosphere) and the velocity of the object. Vehicles transitioning to space usually maintain a constant dynamic pressure to ensure proper rocket engine performance and to maintain structural integrity. The nearly vertical lines on the right side of the graph represent lines of constant specific energy. Specific energy is the sum of the potential and kinetic energy per unit mass. These lines are useful for determining the Gryphon's proximity to a specific orbit and for reentry. After a retro burn (in preparation for reentry), the Gryphon has a certain amount of energy. The retro burn puts the Gryphon in a trajectory that will intersect the Earth. On this trajectory, the Gryphon gradually converts potential energy (height) into kinetic energy (speed), but its total energy remains a constant. If the atmosphere were neglected, the spaceplane would follow one of the constant energy lines down until it hit the Earth. However, as the altitude decreases (and the atmospheric density increases), atmospheric drag slows the spaceplane, and it will start deviating from the constant energy line and begins to run parallel to lines of constant dynamic pressure. The lines in the lower left portion of the graph are lines of equilibrium wall temperature. These are used to give an

indication of the hull temperature of the Gryphon during reentry. If the Gryphon takes a reentry

trajectory too steep, the hull temperature may exceed safety limits and burn up.
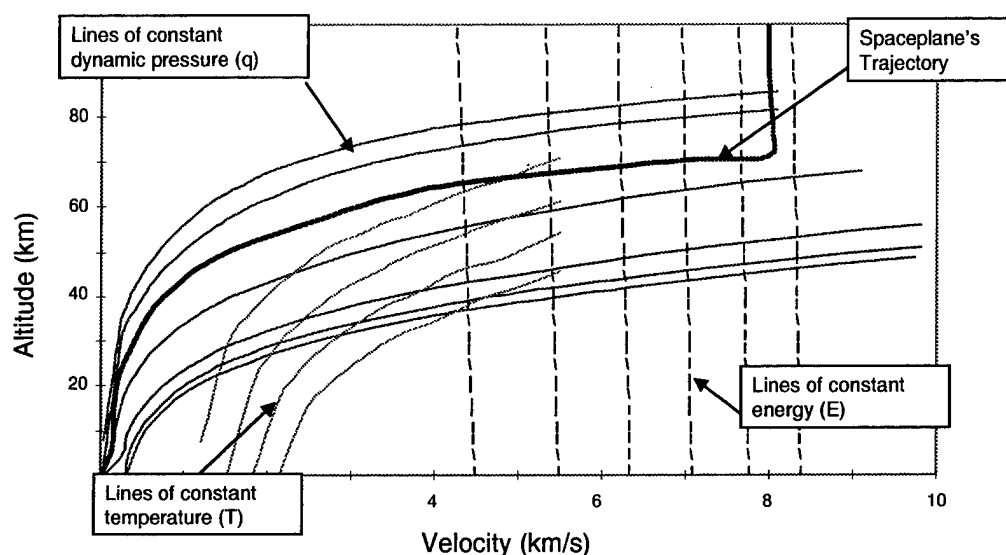
Figure 19 – Exo-atmospheric trajectory in the altitude vs. velocity domain

The routine produces another table of output values that includes time, downrange, velocity,

altitude, alpha, gamma (angle between ground and flight path in radians), weight, thrust, fuel flow,

dynamic pressure (q), and axial acceleration (G's) of the resulting path.

The position of the spaceplane in the environment is determined using the values of

downrange distance (the horizontal distance from the starting point) and altitude.

ORBITENTRYPROP converts these values into a position in WGS84 coordinates for use in the

simulation.

To provide a more realistic orbit entry, once the Gryphon ascents to an altitude of

approximately 70 kilometers using ORBITENTRYPROP, the Gryphon then executes an orbital

41

maneuver called a Hohmann transfer to transition to the desired orbital radius (see Figure 20). This procedure is the same one executed by the NASA Space Shuttle to achieve low-earth orbit. Although there are an infinite number of paths for such a transition, the Hohmann transfer requires the least fuel to achieve the new orbit. Similar to other orbital maneuvers, the Hohmann transfer requires two delta-v burns. To transition from a low orbit to a high orbit, the first delta-v increases the orbiter's speed to place it in an elliptical transfer orbit that is tangent to both the low orbit and high orbit. The final delta-v, performed at the apogee of the transfer orbit, increases the orbiter's speed to achieve the desired orbit.
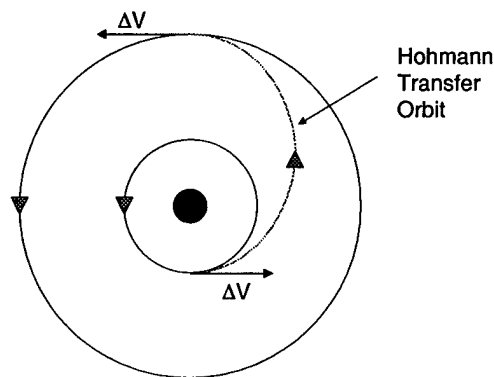
Figure 20 – Hohmann transfer

Although this example discusses the transition from a low to a high orbit, the same principles may be applied to a transfer in the opposite direction for use during reentry. The only difference would be that two decelerations would be required instead of two accelerations.

TaxiProp

Simulating a hard runway used for taxiing and takeoff was accomplished using a technique called intersection testing. A line segment from the Gryphon to the center of the Earth is checked

for an intersection with the polygons making up the surface of the Earth. The distance from the Gryphon to the intersection point is, in effect, the altitude above ground level (AGL). For runway operations, the center of the Gryphon must maintain a fixed distance (the VSP uses 4.5 meters) above the surface of the runway. The illusion of a hard runway is maintained by forcing the altitude AGL to remain constant. When on the runway, the Gryphon is moved straight ahead (defined by its orientation) by a distance determined by the frame rate and the spaceplane's speed. If the runway is not perfectly level, the Gryphon may now have a different altitude AGL than before the movement. The position of the Gryphon is therefore compensated either up or down to maintain a fixed altitude AGL. The resulting propagation model, termed TAXIPROP, is used for all runway operations.

Other Propagation Models

In addition to the propagation models for the Gryphon discussed above, models were developed for the other SIMOBJECTS in the VSP. The VSP uses the center of the Earth as the origin of the simulation. Therefore, the Sun is modeled as revolving around the Earth using the SUNPROP propagation model. The Earth is rotated about its axis of rotation by the EARTHPROP model. The Moon also revolves around the Earth using MOONPROP in this earth-centric simulation. All three of these propagation models were derived from routines developed in the Solar System Modeler [WILL96]. It is interesting to note that, although the earth-moon distance is modeled accurately, the Sun has been placed just beyond the orbit of the Moon (and scaled appropriately). This approach reduces the required size of the viewing frustum, aiding in the elimination of an undesirable rendering property called flimmer. See Lt Rothermel's thesis for other techniques used to eliminate flimmer from the VSP display [ROTH97].

43

Rendezvous

One of the requirements of the VSP is to provide a method for the spaceplane to rendezvous and co-orbit with a satellite or space station. The target satellite presents a moving objective for the Gryphon to attain. For the spaceplane to rendezvous with the target, the VSP first must establish the time of the encounter, then estimate the future position of the target, and finally, determine a path to reach the rendezvous point (see Figure 25).

The first step in executing a rendezvous is to establish the time of the rendezvous. The time specifies the future position of the target and the spaceplane, as well as the fuel requirements for the maneuver. Depending on the urgency of the maneuver, the pilot may wish to rendezvous in the least amount of time, or may wish to use the least amount of fuel, or some option between the two extremes. The spaceplane can, to a degree, trade fuel for time. In other words, the spaceplane can expend additional fuel to reach the target faster. The spaceplane pilot also has the option to wait a period of time before initiating a rendezvous maneuver. Allowing for a wait time allows the spaceplane (and the target) to continue in their current orbits, potentially reducing the fuel required for the rendezvous. The rendezvous time equals the current time plus the wait time and the flight time (the time between the initial and final delta-v burns).

$$t = t_0 + WaitTime + TimeOfFlight \qquad (1)$$

To determine the fastest path to rendezvous, we set the wait time to zero, and incrementally search for the lowest time that will not cause an Earth impact. These steps are presented in Figure 21.

```
mintime ← 0

done ← false

while done = false

    mintime ← mintime + INCREMENT_VALUE

    fuelreq ← fuel requirements for rendezvous.

    if rendezvous trajectory doesn't impact earth

        done ← true
```

Figure 21 – Algorithm for determining lowest time-of-flight for rendezvous .

Although the quickest time-of-flight can be determined rapidly, calculating the cheapest

(least amount of delta-v expended) rendezvous time is not as simple. The cheapest route often calls

for a wait time. The VSP determines the wait-time/time-of-flight combination that minimizes the

delta-v by using a direct double "for-loop" search of wait-times and times-of-flight (see Figure 22).

To allow the search to complete quickly, MAX_RENDEZVOUS_TIME is set to approximately 5½

hours.

```
minfuel ← ∞
for waittime ← 0 to MAX_RENDEZVOUS_TIME
    find position of spaceplane and target after waittime.
    for timeofflight ← 0 to MAX_RENDEZVOUS_TIME - waittime
        fuelreq ← fuel requirements for rendezvous.
        ensure rendezvous trajectory doesn't impact earth.
        if fuelreq < minfuel
            minfuel ← fuelreq
            rendezvous_waittime ← waittime
            rendezvous_timeofflight ← timeofflight
```

Figure 22 – Algorithm for determining least-fuel time-of-flight for rendezvous

Once the time of the rendezvous has been established it is possible to estimate the future position of the spaceplane and the target using a straightforward application of the Kepler procedures used in the AstroProp model described above. The Gryphon and the target move along their current orbits during the wait time. The location of the target at the time of rendezvous is the desired rendezvous point. After the initial delta-v burn, the Gryphon moves along its intercept orbit to meet the target at the rendezvous point.

To accomplish the change in orbit required to meet up with the target, the spaceplane must execute two delta-v burns. The first burn changes the orbit of the spaceplane to bring it into an intercept orbit. Once the spaceplane has reached the rendezvous point, the spaceplane makes a final delta-v burn to achieve the same orbit as the target. The magnitude (and direction) of the two delta-v's are calculated using a method developed by the German mathematician, Carl Fredrich Gauss. Gauss' method allows for the calculation of an orbit (in our case, the transfer orbit) given two position vectors, r_1 and r_2, and the time-of-flight, t, between them. Although there are an infinite

number of orbits which pass through points r_1 and r_2, only two have the specified time-of-flight, one for each direction of motion (see Figure 23). The short trajectory is always less than π radians, whereas the long trajectory is always more than π radians.
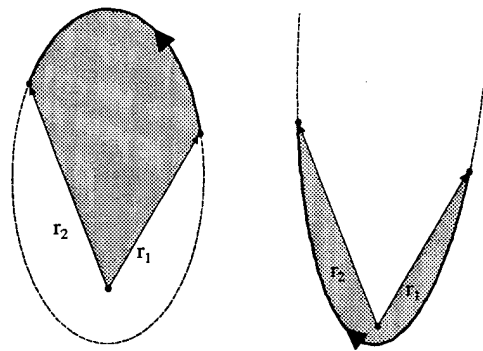
Figure 23 – Short and Long way trajectories with the same time of flight

The solution to the Gauss problem involves solving a series of three transcendental equations. Bate, et al, provides a detailed discussion of several methods for solving the equations, all using the trial-and-error technique necessary when solving transcendental equations [BATE71]. In the VSP, the solution to the Gauss problem is computed using a procedure in the astrodynamics FORTRAN library obtained from the Air Force Academy. The general method of solving the Gauss problem is shown in Figure 24:

47

```
1) Guess a trial value of one of the three
   unknowns.
2) Using Gauss' equations, compute the
   remaining two unknowns.
3) Test the result by solving for t, and check
   against the given value of time-of-flight.
4) If the computed value of t does not agree
   with the given value, adjust the value of
   the variables and repeat the procedure
   until it agrees.
```

Figure 24 – General method for solving Gauss' problem for rendezvous calculation

Finally, for the transfer orbit to be useful, the spaceplane must not impact with the Earth on its way to the rendezvous point. Therefore, before the orbit is presented to the pilot as an option, the VSP checks the orbit to ensure the Gryphon's flight path remains out of the Earth's atmosphere. Figure 25 shows a potential transfer orbit to rendezvous with a DMSP satellite.

Figure 25 – VSP's Orbit Display showing a potential rendezvous

SimGryphon

To simulate the capabilities of the MSP, the ability to use multiple propagation models, as well as operate its payload and landing gear systems, were added to the Gryphon. The Gryphon (the virtual vehicle traveling through the virtual environment) is modeled as a subclass of SIMOBJECT (described above on page 30) called SIMGRYPHON. The relationship of the SIMGRYPHON to the rest of the simulation software is shown in Figure 26. The SIMGRYPHON has the responsibility for maneuvering through all the environments the MSP will encounter. SIMGRYPHON also has special methods for handling the spaceplane's accessories (e.g., payload doors and landing gear), as well as for deploying a satellite. At the beginning of the simulation, the spaceplane can begin on the runway or in low-earth orbit (using command line options) by calling SIMGRYPHON'S `InitializeToTaxiProp` or `InitializeToSpace`. The `GetAAGL` method is used to determine the spaceplane's altitude above-ground-level. SIMGRYPHON'S

remaining methods are used to control the spaceplane's landing gear and payload bay doors, and for

preparing the payload for deployment.
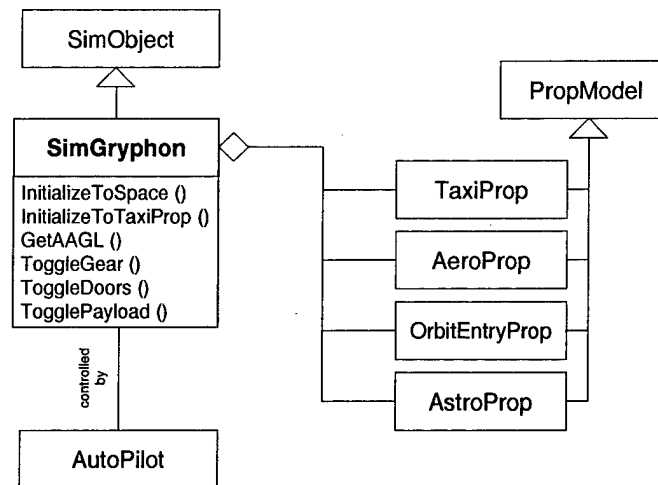
Figure 26 – Relationship of SimGryphon to other objects

Modifying the appearance of the spaceplane

One of the responsibilities of SimGryphon is to model the spaceplane's landing gear and

payload doors. To model these movable parts, we first created a complete model of the spaceplane,

then saved the gear and doors separately to allow each object to have independent motion (see

Figure 27).
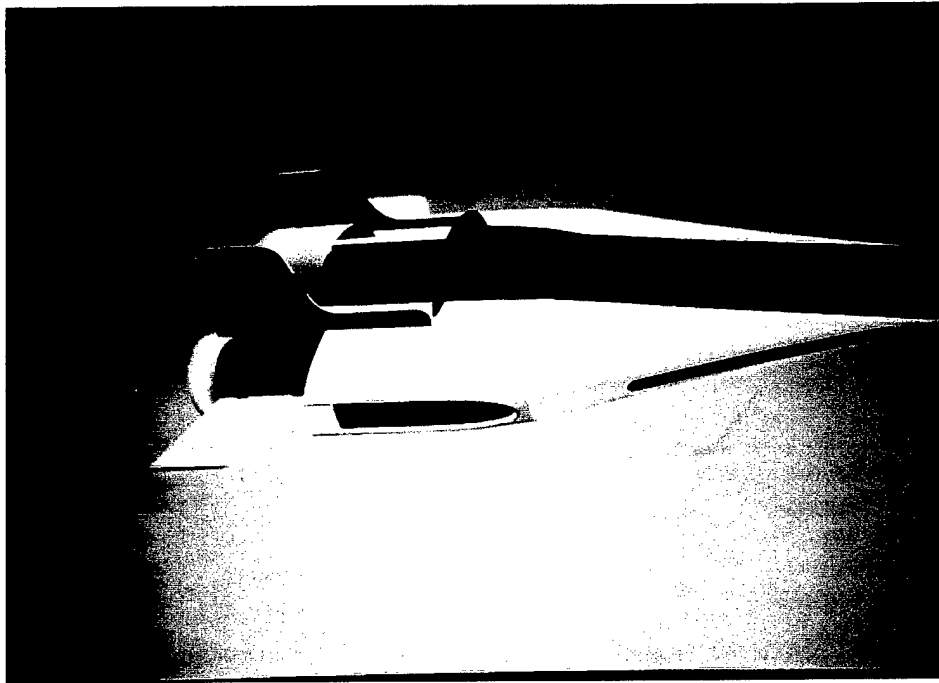
Figure 27 – The payload bay doors are created separately from the spaceplane

The adjustable payload doors allow the spaceplane to deploy a satellite. Prior to deployment, the satellite is modeled as a movable part of the SIMGRYPHON, similar to the landing gear. When it is ready to be deployed, the satellite is removed from the SIMGRYPHON coordinate system and initialized as a new SIMOBJECT with its own propagation model. Once it has been deployed, the satellite moves independently of the Gryphon. One of the methods used in the Space Shuttle for ejecting its payload is through a spring that pushes the satellite out of the payload bay. The VSP accomplishes this capability by applying a small delta-v to the satellite just as the new SIMOBJECT is initialized. Figure 28 shows the satellite shortly after it was deployed from the spaceplane.
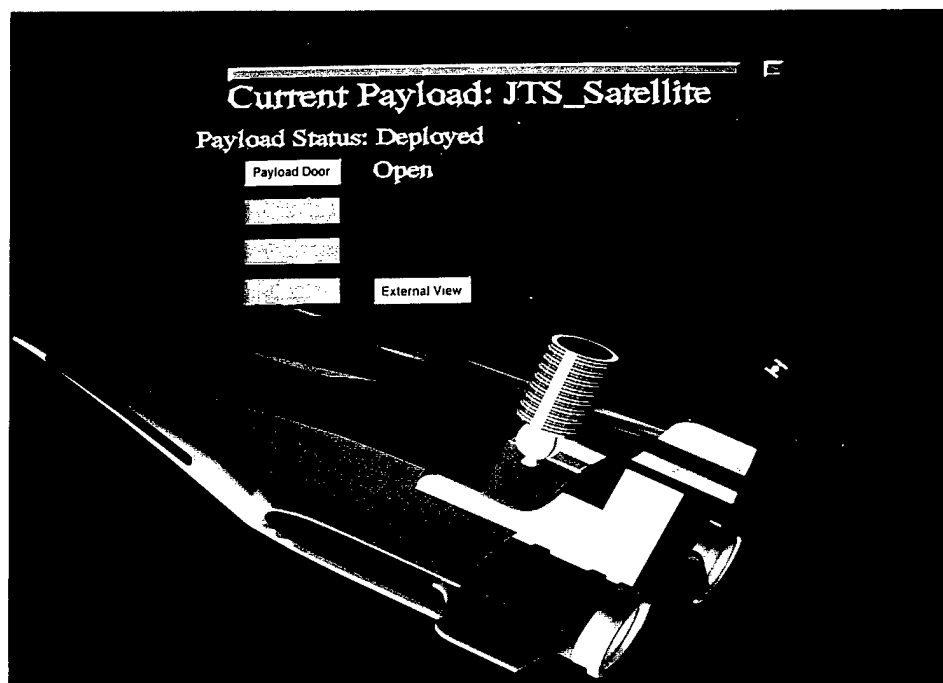
Figure 28 – A display of the deployed payload with the payload doors open

Transition between propagation models

Ideally, the SIMGRYPHON would use a single propagation model to manage its motion through the environment. However, no single model was found that was capable of accurately modeling the spaceplane's movement in the broad spectrum of environments from runway to space. Each propagation model specializes in a single domain. For this reason, the SIMGRYPHON was designed and developed to allow the ability to change its propagation model as it transitions from one domain to another. Many of the propagation models discussed above were developed specifically for the SIMGRYPHON. Figure 29 shows all of the propagation models used by the spaceplane.

52
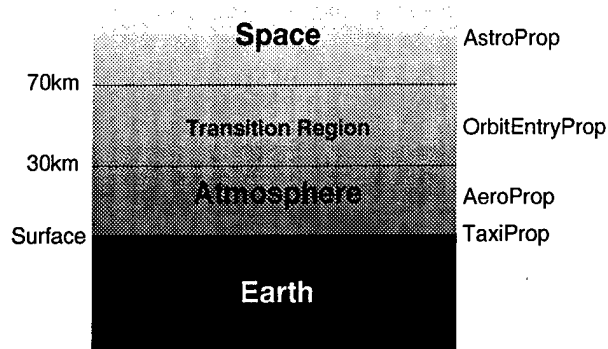
Figure 29 – The spaceplane's propagation models

Although the WGS84 system is the primary coordinate system used in the VSP, each propagation model operates in its own coordinate system. It is therefore necessary to provide the capability to convert between the coordinate systems (see Chapter Two for a summary of the coordinate systems used in the VSP). Coordinate conversion becomes especially important when transitioning from one propagation model to another. The section entitled Coordinate Conversion below details the procedures used to convert between the VSP's many coordinate systems.

The changeover from one propagation model to another must be accomplished such that the transition is not obvious to the user. For example, during takeoff, the transition from TAXIPROP to AEROPROP is performed when the Gryphon is moving fast enough and has pitched its nose up. The Gryphon's position and orientation are converted from the WGS84 coordinate system used by TAXIPROP into the NED coordinate system used by AEROPROP. A second concern during the takeoff transition is the Gryphon's speed; the AEROPROP'S speed is initialized using units of *mach* rather than meters per second as used by TAXIPROP. Although the speed of sound (mach) is dependent on air density (and therefore altitude and temperature), the VSP assumes the speed of sound to be 343.0 m/s for the AEROPROP to TAXIPROP transition. Each transition from one

propagation domain to the next requires a similar conversion of position, orientation, and velocity (see Figure 30).

```
move spaceplane using current propagation model.
switch (current propagation model)
    :

    case AeroProp:
        if altitude_AGL < LANDING_ALTITUDE
            convert current position from ENV to WGS84 for
                TaxiProp
            initialize TaxiProp with WGS84 position,
                orientation and velocity.
            current propagation model ← TaxiProp
            begin automatic application of brakes.
        if altitude > ORBIT_ENTRY_ALTITUDE
            convert current position from ENV to WGS84 for
                OrbitEntryProp
            initialize OrbitEntryProp with WGS84 position,
                orientation and velocity.
            current propagation model ← OrbitEntryProp
    case OrbitEntryProp:
    :
```

Figure 30 – Algorithm used by SIMGRYPHON to change propagation models

Coordinate Conversion

The VSP must be capable of operating in a variety of coordinate systems; some are used for convenience, while hardware, software and operational requirements dictate others. A description of the many coordinate systems used in the VSP is located in Chapter Two. This section will describe a means of converting between them.

While investigating methods of coordinate conversion, we first looked at the Round Earth Utilities, written for previous AFIT virtual environments. The Round Earth Utilities (REU) were designed to provide routines for converting between the flat-earth ENV coordinate system and the round-earth WGS84 coordinate system [ERIC93]. This conversion was accomplished by placing a patch of flat terrain near the region of interest as shown (two-dimensionally) in Figure 31, where **a** is the radius of the Earth. The REU did not, therefore, consider the curvature of the Earth. The figure shows that as the distance from the flat-earth origin increases, the error between the REU computed position and the actual position increases. The REU also considers the Earth to be a sphere. Because the Earth is more accurately modeled as an ellipsoid, treating the Earth as purely spherical will contribute to position errors when used close to the poles. The existing REU also did not provide the capability to use geodetic coordinates or ECI coordinates.
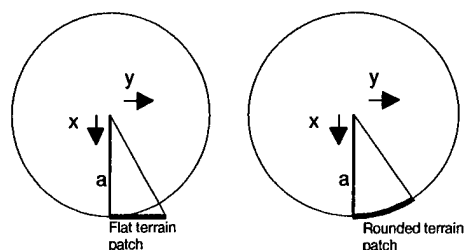
Figure 31 – Problem with the previous version of Round Earth Utilities

Because of these difficulties, the decision was made to rewrite the REU. However, this decision brought on the question of computer platform independence that is desired, which boils down to a decision on whether to use Performer's math library. The REU makes use of the Performer matrix mathematics library. Performer provides an extensive library of matrix and

vector math routines useful when converting between two coordinate systems. The Performer math libraries are optimized for operation on Silicon Graphics hardware.

A desire to remove the Performer math dependencies from the REU was noted in a prior project [ZURI96]. Basing the conversion utilities on the Performer math routines works well for many of the VE applications developed at AFIT because the applications are already using the Performer libraries for graphics support. However, some previous (and current [HUTS97]) applications do not use Performer's graphics routines, but need the coordinate conversion capabilities of the REU. Requiring a program, that would not otherwise do so, to load the Performer libraries unnecessarily increases the program size and load time. Linking the coordinate conversion utilities to Performer also limits the application to operation on Silicon Graphics hardware.

Besides deciding which math library to use, a second design decision had to be made regarding how to package the new coordinate conversion utilities. The first option was to emulate the REU and design our new utilities as a C++ class. A class provides a convenient place to hold data as well as the methods that use the data. On the other hand, creating a class involves additional overhead. A class is unnecessary if all data required by the functions are passed in as parameters. In such cases, it may be more practical to avoid the overhead of a class and design the utilities as a collection of stand-alone routines.

Given the difficulties with the Round Earth Utilities, the decision was made to rewrite the Round Earth Utilities as a stand-alone set of routines that do not use the Performer math libraries. I shall refer to the new routines as the AFIT Coordinate Conversion Utilities (ACCU). Figure 32 identifies the coordinate conversion routines provided by the ACCU.

```
 ┌─────────────────────────┐
 │          ACCU           │
 ├─────────────────────────┤
 │ WGS84 to ENV ()         │
 │ ENV to WGS84 ()         │
 │ WGS84 to ECI ()         │
 │ ECI to WGS84 ()         │
 │ WGS84 to Geodetic ()    │
 │ Geodetic to WGS84 ()    │
 │ Performer to DIS ()     │
 │ DIS to Performer ()     │
 │ WGS84_Delta ()          │
 └─────────────────────────┘
```

Figure 32 – Coordinate conversion routines in the ACCU

The ACCU routines make use of an algorithm described by Ralph Toms for converting

between a geocentric (WGS84) coordinate system and a geodetic (latitude/longitude/altitude)

system [TOMS93]. The conversion from geodetic to geocentric coordinates is straightforward.

The inverse transformation is performed using an efficient iterative algorithm. Toms' coordinate

conversion algorithms are outlined in Appendix B.

Conversions to and from the NED flat-earth coordinate system are not directly addressed in

the ACCU. When a conversion from WGS84 to NED coordinates is needed, the WGS84 position

is first converted to the ENV system. The conversion from ENV to NED is elementary as shown in

Equation (2).

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{ENV} = \begin{bmatrix} y \\ x \\ -z \end{bmatrix}_{NED} \qquad (2)$$

Because the Aero Model is a flat-earth based model and the VSP uses the round-earth

WGS84 coordinates as its primary coordinate system, a coordinate conversion between a flat-earth

coordinate system and a round-earth coordinate system is necessary.

The conversion from a position in the WGS84 coordinates to the ENV coordinate system (round-to-flat) uses the following procedure:

1. Convert from geocentric (WGS84) into geodetic coordinates using Toms' geocentric-to-geodetic routine.

2. Assume a fixed distance between lines of latitude and longitude[1] (1 degree ≈ 111.319 km). This assumption flattens the Earth in a manner similar to a Mercator map projection.

The conversion from a position in the ENV coordinate system to the WGS84 coordinates (flat-to-round) uses the following procedure:

1. Convert from ENV into a geodetic coordinate system by assuming a fixed distance between lines of latitude and longitude (similar to step 2 above).

2. Use Toms' algorithm to convert from geodetic into geocentric (WGS84) coordinates.

In addition to the position, an object's orientation must also be converted from one coordinate system to another. Conversion of orientation between round and flat earth coordinates becomes complicated because the definition of "straight up" in the ENV system is in the positive z-axis, but in the WGS84 system the definition of "straight up" depends on location on or above the Earth's surface. The two coordinate systems are coincidental only at the North Pole.

To convert the orientation of an object between the two coordinate systems, we must first establish a reference vector that points in a fixed direction regardless of the object's location on Earth. The ACCU accomplishes this conversion by calculating the vector that points north given the WGS84 position of the object. Notice in Figure 33 that the North vector points in various directions depending on location. A similar method was used in the Solar Modeler for orienting

---

[1] Some difficulties arise due to this assumption, and will be addressed in the section entitled Flat-to-Round Drift below.

satellites toward the Earth's surface [WILL96]. It is then necessary to "add in" the flat-earth orientation to obtain the round-earth orientation. Alternately, to obtain the flat-earth orientation, it is necessary to "subtract out" the round-earth orientation from the North vector.
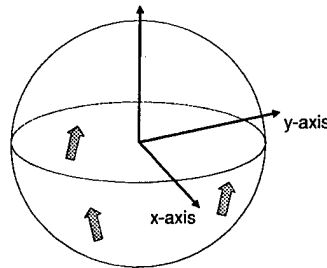
Figure 33 – A depiction of three North vectors.

We can use Euler angles to represent the orientation of the object. Each of the Euler angles represents rotation about one of the three primary axes. From the Euler angles, we can compute an Euler matrix, E:

$$E = R \cdot P \cdot H \tag{3}$$

where R, P, and H represent the roll, pitch and heading transform matrices respectively. Euler angles and matrices are explained in more detail in Appendix A.

The Euler matrix for the round-earth orientation, E_R, is computed by multiplying the flat-earth Euler matrix, E_F, by the North vector Euler matrix, E_N:

$$E_R = E_F \cdot E_N \tag{4}$$

The Euler matrix for the flat-earth orientation is therefore:

$$E_F = E_N \cdot [E_R]^{-1} \tag{5}$$

Once the Euler matrix in the desired coordinate system has been computed, the next step is to extract the required heading, pitch and roll information from the matrix. Performer provides a routine, `getOrthoCoord`, which accomplishes the Euler angle extraction automatically [SGI95]. However, one of our goals was to design the ACCU without the use of the Performer's math

59

routines. The same information provided by getOrthoCoord can be obtained without the assistance of Performer using a procedure described by Thomas [THOM91]. Appendix A describes Thomas' method used to extract the orientation information from an Euler matrix.

Flat-to-Round Drift

One of the primary uses for the ENV-to-WGS84 (flat-to-round) coordinate conversion discussed above was to allow the VSP to use the Aero Model (also discussed previously). The Aero Model was designed to operate in a flat-earth coordinate system, whereas the VSP uses the WGS84 round-earth coordinate system. The AEROPROP propagation model converts from the flat-earth system to round-earth coordinates for use in the simulation. However, during the development of the VSP, we recognized a weakness in the procedure described above to convert the position from the Aero Model into round-earth coordinates.

As discussed in the previous section, to convert from ENV to WGS84 coordinates, the ACCU first converts from ENV into geodetic coordinates. The conversion to geodetic coordinates assumes a fixed distance between lines of latitude and longitude. A quick glance at a globe, however, confirms that the distance between lines of longitude decreases as you approach the poles. The ACCU's ENV-to-WGS84 conversion doesn't behave as expected when we use the Aero Model output, as the following example demonstrates.

For the sake of this example, assume the spaceplane begins at 35°N, 118°W (near Edwards AFB) flying northeast. The spaceplane travels a distance such that, in the Aero Model's flat-earth coordinate system, it has traveled 1km north and 1km east (see Figure 34). Using the ACCU's assumption of a fixed distance (111.319km) per degree, the spaceplane traveled 8.98×10^{-3} degrees in both the north direction and east direction. Lines of longitude converge as the cosine of the latitude. Thus, near Edwards AFB, lines of longitude are separated by only 91.187km per degree.

If we compensate for the convergence of lines of longitude, the spaceplane actually traveled 8.98×10^{-3} degrees north, but 10.97×10^{-3} degrees east. Thus, the assumption of a fixed distance between lines of longitude forces the spaceplane to fly slower to the east than it should. The spaceplane therefore appears to drift in the north-south direction.
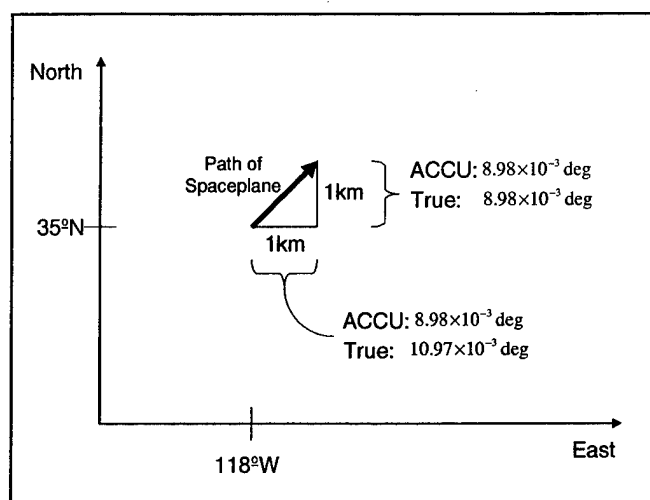
Figure 34 – Flat-to-Round drift problem

To solve this problem, a routine was added to the ACCU to compensate for the spaceplane's latitude. The new routine, WGS84_Delta, when given the spaceplane's current WGS84 position and the change in ENV coordinates (provided by the Aero Model), returns the spaceplane's position in WGS84 coordinates. The new "delta" routine is most simply represented in geodetic coordinates as outlined in Figure 35. For any other coordinate system, simply convert the geodetic results to the desired coordinate system.

1) A constant distance separates lines of latitude. Therefore, first compute the change in latitude based on distance traveled in the northerly direction.

2) Using the current latitude, determine the distance between lines of longitude:

$$Meters\_per\_degree = \frac{Radius\_of\_earth \cdot \cos(latitude)}{180/\pi}$$

3) Using the result of step 2, compute the change in longitude based on distance traveled in the easterly direction.

4) Compute the new latitude and longitude by adding the results of steps 1 and 3 to the current latitude and longitude.

Figure 35 – Algorithm to compensate for flat-to-round drift

Auto-Pilot

Most of the maneuvering done in the Spaceplane will be performed automatically because of the complexity of operating a vehicle in space. The operation of the Spaceplane will be a collaboration between the human pilot and the orbiter with the pilot in the position of flight manager and the onboard computer tasked with accomplishing assigned tasks. For example, rather than manually firing a precise series of short thruster bursts, the pilot may instead specify a new orbit and allow the computer to accomplish the maneuver. This paradigm has several advantages, key of which is the elimination of menial tasks from the pilot. The pilot is free to think, "Where do I want to go?" rather than worrying about "How do I get there?".

Figure 36 identifies the methods provided by the AUTOPILOT to automatically maneuver the spaceplane. When the AUTOPILOT is active, the Fly method is called once per frame, providing the

autopilot with a chance to observe the environment (e.g., current altitude, distance from waypoint) and make any necessary maneuvers to the spaceplane. The `SetState` method is used to change from one state to another (see Figure 37 for the AUTOPILOT'S state diagram). For example, when beginning an automated landing, `SetState` would be called to transition from FLYROUTE to LANDING. The remaining methods specify the activity the AUTOPILOT is to perform. `AssignTakeOffRoute`, `AssignAeroRoute` and `AssignLandingRoute` are used to specify the waypoints for AUTOTAKEOFF, FLYROUTE, and AUTOLAND. `AssignOrbitChange` is used to specify a new orbit while in space, and `AssignHohmann` is used during the transition to low-earth orbit.

AutoPilot
Fly ()
SetState ()
AssignTakeOffRoute ()
AssignAeroRoute ()
AssignLandingRoute ()
AssignOrbitChange ()
AssignHohmann ()

Figure 36 – Methods available in AUTOPILOT

The interface will provide the pilot with the ability to choose automatic execution of almost all portions of the mission, including automated takeoff, route following, orbit entry, and landing. Because of the radically different propagation models used, the VSP autopilot is best described as three parts: the atmospheric autopilot, the space autopilot and the transition autopilot (see Figure 37). To allow the autopilot to adjust its activities as it moves from one flight regime to another, the AUTOPILOT must determine the propagation model currently used by the spaceplane. To accommodate the AUTOPILOT'S need to know the current region of operation, each PROPMODEL provides a `GetPropModelType` method (described above).
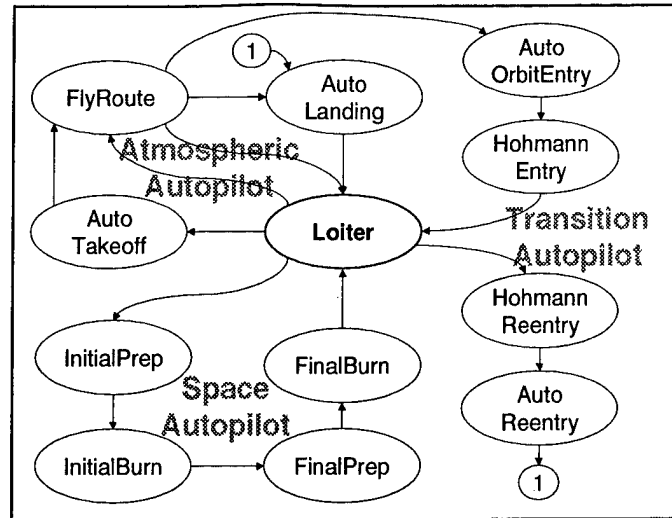
Figure 37 – AUTOPILOT state diagram

Atmospheric Autopilot

Many modern aircraft have some form of autopilot to assist the pilot. The degree of automation performed by the autopilot varies from aircraft to aircraft, but a commonly automated task is the ability to follow waypoints along the desired route. The VSP atmospheric autopilot uses the `Fly` routine as a basis for most of its capabilities, including AUTOTAKEOFF, and AUTOLAND, and FLYROUTE.

When the pilot activates the autopilot inside the atmosphere, the AUTOPILOT changes to the FLYROUTE state, and the VSP follows a predefined route. This capability was built from the same package of code as the Aero Model. The waypoints are specified using the `AssignAeroRoute` method. Each waypoint is defined by its position (i.e., latitude, longitude and altitude), by the speed the spaceplane should be travelling when it reaches the waypoint, and by how close the spaceplane must pass by the waypoint to consider the waypoint achieved. The autopilot maneuvers toward the waypoint by adjusting the throttle and control surfaces just as a human pilot does. The waypoints used by the spaceplane in the atmosphere are read in from a data file.

64

While on the runway, the pilot also has the option of requesting an AUTOTAKEOFF. When this option is selected, the VSP applies full throttle and tries to nose up the spaceplane. Once it has achieved a safe flight speed, the spaceplane lifts off the runway. It then automatically begins following a route high in the atmosphere, where it transitions into the AUTOORBITENTRY mode and begins the journey to space.

When finished with the space mission, the pilot may choose an automated landing. In the AUTOLAND mode, the autopilot follows a set of waypoints designed to provide a safe landing.

Space Autopilot

Because maneuvering in space is considerably different than in the atmosphere, the method used for following waypoints is not appropriate for space flight. While developing the space autopilot, we first analyzed the required mission profiles for the VSP. The primary use of the autopilot in space is to allow the Spaceplane to match the orbit of another satellite. The INITIALPREP, INITIALBURN, FINALPREP, and FINALBURN autopilot capabilities were developed to meet this need.

As described previously on page 44, a rendezvous maneuver requires two accurately timed applications of delta-v. The exactness required to perform such a procedure makes low-earth rendezvous an excellent candidate for automation. When the pilot is planning a rendezvous, the interface presents the pilot with fuel-consumption/time tradeoffs for three different orbits to accomplish the maneuver, allowing the pilot to choose whether it is more appropriate to reach the target quickly or to conserve fuel. Once the pilot has chosen the appropriate fuel/time tradeoff for the current situation, the autopilot takes over to execute the maneuver. In preparation for the initial delta-v burn, the VSP initiates a pre-burn orientation change to ensure the exhaust of the spaceplane faces in the correct direction. After the initial delta-v burn, there will be a wait period before the

final delta-v burn. Immediately before the final delta-v burn, the spaceplane executes another pre-burn orientation change. The final delta-v burn brings the spaceplane into a co-orbit with the target.

Transition Autopilot

The primary means of orbital entry in the Virtual Spaceplane is using the autopilot. This procedure is similar to the automated liftoff procedures used by today's space shuttle. Because of its nature, the automated orbit entry procedure is important both in the atmospheric as well as space portions of the autopilot. When the spaceplane begins its flight toward space, the atmospheric autopilot follows a route (using FLYROUTE) bringing the spaceplane high enough for the transition autopilot to take over. The autopilot then changes its state to AUTOORBITENTRY.

The AUTOORBITENTRY and AUTOREENTRY procedures work in concert with the atmospheric part of the autopilot. The AUTOORBITENTRY function provides an automated method for achieving a low-earth orbit. The atmospheric autopilot handles the first part of this maneuver by flying the spaceplane into the upper atmosphere. The autopilot handles the transition through the tenuous upper atmosphere as described above in the ORBITENTRYPROP section. Once the VSP reaches an altitude where orbital mechanics models (and therefore the ASTROPROP propagation model) operate effectively and accurately, it executes a Hohmann transfer to bring the spaceplane into low-earth orbit.

Atmospheric reentry works similar to the orbit entry procedure described above, only in reverse. However, the initial Hohmann transfer must be accurately timed to bring the spaceplane into atmospheric reentry for a safe landing at the designated landing site. Once the autopilot has performed the Hohmann transfer and brought the spaceplane through the upper atmosphere, it automatically transitions into the AUTOLAND mode to provide a safe, automated finish to the mission.

Hypertext Interface

One of the primary research goals identified by the thesis statement was to investigate new concepts in cockpit design. We looked to hypertext as an option in a cockpit in which, without proper design, the pilot can potentially be overwhelmed with data. Therefore, because hypertext provides a rapid, convenient means of locating significant information, the Hypertext Markup Language (HTML) was investigated for use in the user interface. However, integration of a standard HTML browser into the VSP is not straightforward because browsers are designed to be used in a standard screen windowing environment, not in a 3D virtual environment. To overcome this limitation, I examined the options of designing a new browser, integration of an existing browser, and designing hypertext panels off-line and preloading them at program startup to provide a pseudo-HTML hypertext interface.

Design new HTML browser

One approach for integration of a hypertext interface into our virtual environment was to design and implement our own HTML browser capable of displaying HTML pages on a panel in the VSP. This approach does have the advantage of ease of design of new panels. New panels could be put together using the HTML standard language. The new browser could be expanded to include some of the newer capabilities on the Internet, such as Java. This approach also has several disadvantages. Writing (and debugging) our own browser using IRIS Performer would take a significant amount of time. Secondly, the Performer browser would need to be modified whenever a new change to the HTML standard is released.

Integration of external HTML browser

The second approach considered for incorporating an HTML browser into the VSP was to make use of the Performer technique called texturing. Texturing allows a picture to be displayed over geometry in the scene. The plan was to capture the image of an external browser (e.g., Netscape) window running in the background, and display it in the scene as a texture. Mouse events would be passed back to the Netscape window from the VSP causing Netscape to update. This procedure (capture Netscape and pass back mouse events) would continue as long as the HTML browser was active. This approach would allow the integration of a fully functional Internet capable browser into the virtual environment.

Several difficulties with this method were encountered. First, because the screen captures would be written to a file, Performer would pause as it was loading in the new Netscape texture from the file. The frame rate of the simulation would therefore suffer noticeably. The second, and more significant, problem was capturing the contents of the Netscape browser running in the background. A simple screen capture could work, but the browser window would have to be unobstructed. The Netscape window would therefore obstruct a portion of the Spaceplane virtual environment. One of the goals of any immersive virtual environment like the VSP is to provide the user with a sense of "presence", the feeling that the user is within the VE and behaves toward it as though it were real [ZELT92]. Obstructing the VSP display impairs this sense of presence, and is therefore not acceptable.

Potential workarounds to the obstruction problem included purchasing a second frame buffer for the machine. The additional frame buffer would allow the Netscape window to be "unobstructed" on the secondary frame buffer while the primary frame buffer was used for the VSP. Another workaround would require Netscape to run on a second machine, which would capture the

image and route it across the network to the VSP. Because of the desire for the VSP to run without specialized hardware, these workarounds were deemed impractical.

Preloaded Images

To provide some form of an hypertext interface, a third approach that avoids the difficulties described above was examined: pre-load the necessary Netscape windows and simply switch between them as needed. This approach has the advantage of being fast, but limits us to a small number of HTML pages due to limited texture memory. Because the VSP is being built as a prototype, we selected a few significant pages, designed the images off-line, and then pre-load them at startup.

This "pseudo-HTML" browser provides many of the fundamental features common to a commercial Internet browser. The user can move from page to page by selecting hotspots on the pages. The user can also choose to go "forward" or "back", features familiar to users of Internet browsers. Figure 38 shows an example of the hypertext display in the VSP. However, because the pages are "screen captures" and not described using the HTML language, modifying the pages is more difficult. The VSP's pseudo-HTML pages also require significantly more memory than the equivalent HTML code.
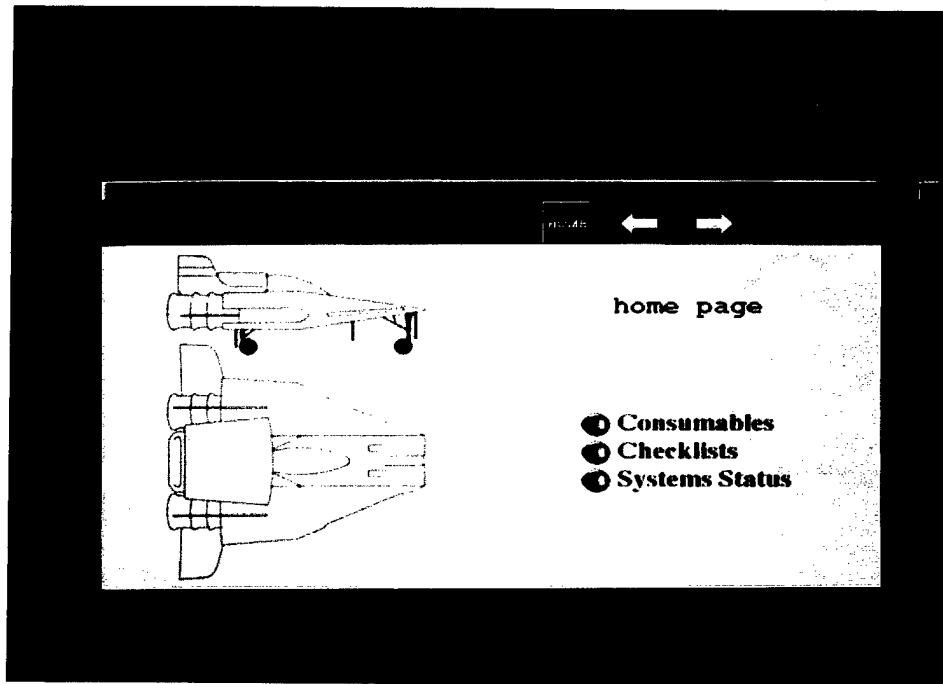
Figure 38 – VSP's hypertext Engineering Panel

The VSP's pseudo-HTML interface is controlled by the HTML Manager (see Figure 39). The Manager manages the HTML panels in pairs, a left panel and right panel. As seen in Figure 38, displaying panels in pairs provides a wider HTML interface that fits well in the spaceplane's console. While developing an HTML panel, the developer may define multiple hotspots. Each hotspot is defined by the page coordinates for the hotspot, and the linked page of information. When a particular hotspot is selected, the HTML Manager displays the corresponding panels of information. Two stacks, the `forward list` and `backward list`, are used to allow the user to step back (and forward) to previously accessed pages of information. The HTML Manager also has a "home page" that is initially displayed and can be recalled by pressing the homepage button on the hypertext interface.

70
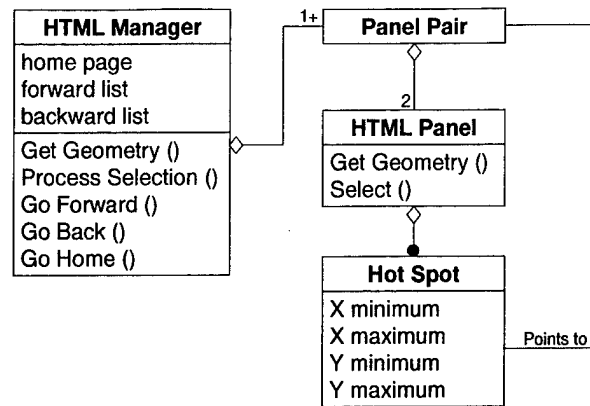
Figure 39 – Pseudo-HTML interface architecture

Summary

In this chapter we covered the research performed during the completion of the Virtual

Spaceplane. The next chapter will recap the requirements identified in Chapter Three, and show

how this work accomplishes the goals of the Spaceplane

71

V – RESULTS

The previous chapter covered the design and implementation of work to fulfill the requirements as outlined in Chapter Three. This chapter will discuss how the VSP requirements were satisfied.

Completion of Requirements

Emulating the style of Chapter Three, the following sections present a summary of the overall VSP requirements and a brief description of how each requirement was satisfied. This synopsis will accompany a more detailed discussion and analysis of requirements completed with regard to this research. Those requirements not directly discussed in this thesis were addressed by the other members of the Virtual Spaceplane team, Capt Lewis and Lt Rothermel [LEWI97][ROTH97].

Simulated Capabilities

The simulated capabilities requirements are aimed at providing a simulation whose capabilities are similar to those of the Military Spaceplane, including accurate flight characteristics, as well as manual and automatic control of the spaceplane operations. Table 8 summarizes how these requirements were addressed.

Table 8 – Completion of Capability Requirements

ID	Requirement	Resolution
Flight Characteristics		
1.11	Maneuvering on runways	TaxiProp
1.12	Flight through the atmosphere	AeroProp
1.13	Maneuvering in space	AstroProp
1.14	Transition between flight regimes	OrbitEntryProp, ACCU, Common interfaces of PropModels
Manual Operation		
1.21	Manually operate in the atmosphere	Input methods of AeroProp
1.22	Manually operate in space	Input methods of AstroProp
Automatic Operation		
1.31	Automatically takeoff	Takeoff mode of Autopilot
1.32	Automatically fly specified route	FlyRoute mode, Route of waypoints
1.33	Automatically enter orbit	EnterOrbit mode of Autopilot
1.34	Automatically modify orbital parameters	Hohmann and Rendezvous modes
1.35	Automatically reenter the atmosphere	Reenter mode of Autopilot
1.36	Automatically land	Landing mode of Autopilot

The previous chapter described how the division between propagation model and geometry model allowed the spaceplane to operate in many flight regimes. The separation of PropModel and SimObject alleviated the need to create a propagation model that operated in all flight regimes. The spaceplane used four propagation models throughout the simulation: TaxiProp for movement on a runway, AeroProp for atmospheric flight, AstroProp for operation in low-earth orbit, and OrbitEntryProp for the transition region between air and space. The transition from one model to another was enabled by the AFIT Coordinate Conversion Utilities, which provides methods for conversion between the many coordinate systems used in the simulation.

The AUTOPILOT class can perform a variety of basic maneuvering operations based on its state (TAKEOFF, FLYROUTE, LANDING, RENDEZVOUS, etc.), the current PROPMODEL, and pilot orders. The GetPropModelType method of PROPMODEL, in conjunction with the AUTOPILOT'S

state, provided the capability to automatically control the Gryphon across multiple flight regimes. For example, the TAKEOFF mode typically started in TAXIPROP, but finished at a safe altitude above the ground in the AEROPROP model. By knowing the PROPMODEL type, the autopilot reacted differently and provided appropriate inputs at different stages of a task. The VSP's autopilot still has several limitations, but can be easily extended during future research.

Supported Missions

The missions listed in Table 9 were successfully integrated into the VSP. The current implementation places many restrictions and/or makes assumptions concerning parameters affecting the missions. For example, the rendezvous mission does not include any time restrictions and the satellite deployment mission does not accurately model the boost to a higher orbit typical of many satellite launches.

Table 9 – Completion of Mission Requirements

ID	Requirement	Resolution
	Supported Missions	
2.1	Rendezvous with orbiting satellite	AutoRendezvous capability
2.2	Deployment of satellite	SimGryphon's payload capability

Because of the difficulty involved with manual maneuvering in space, the capability to rendezvous with a satellite in orbit is provided using the AUTOPILOT. The target panel (Figure 40) allows selection of the target. The figure shows a Defense Meteorological Satellite Program (DMSP) satellite as the current target. Once the target is identified, the pilot can choose from three different intercept trajectories. The first provides the lowest time-of-flight, the second option affords a more fuel-efficient trajectory, while the third option represents the lowest delta-v path. In addition to intercepting satellites, the VSP also rendezvous with the space station.

74
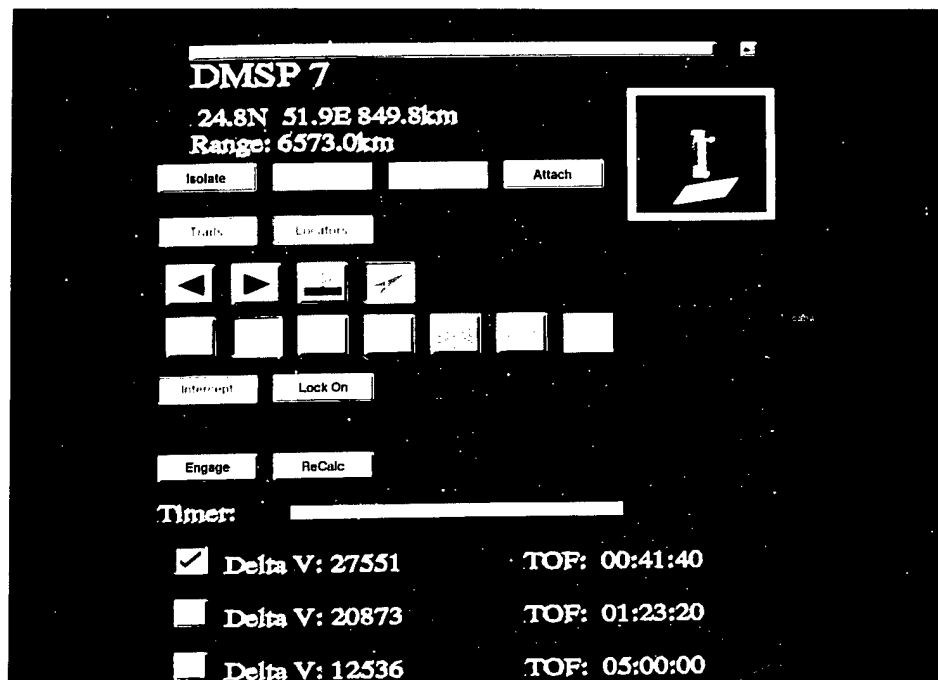
Figure 40 – Display of the VSP Target Panel

The VSP also has the capability to deploy a satellite into a low-earth orbit. The SimGryphon class accommodates the geometry of the satellite inside the payload bay until ready for deployment. Prior to deployment, the pilot can check on the payload's status using the hypertext Engineering Panel. When deployed, the satellite leaves the payload bay like performed with the Space Shuttle. Figure 41 shows the deployed satellite after leaving the payload bay of the spaceplane.
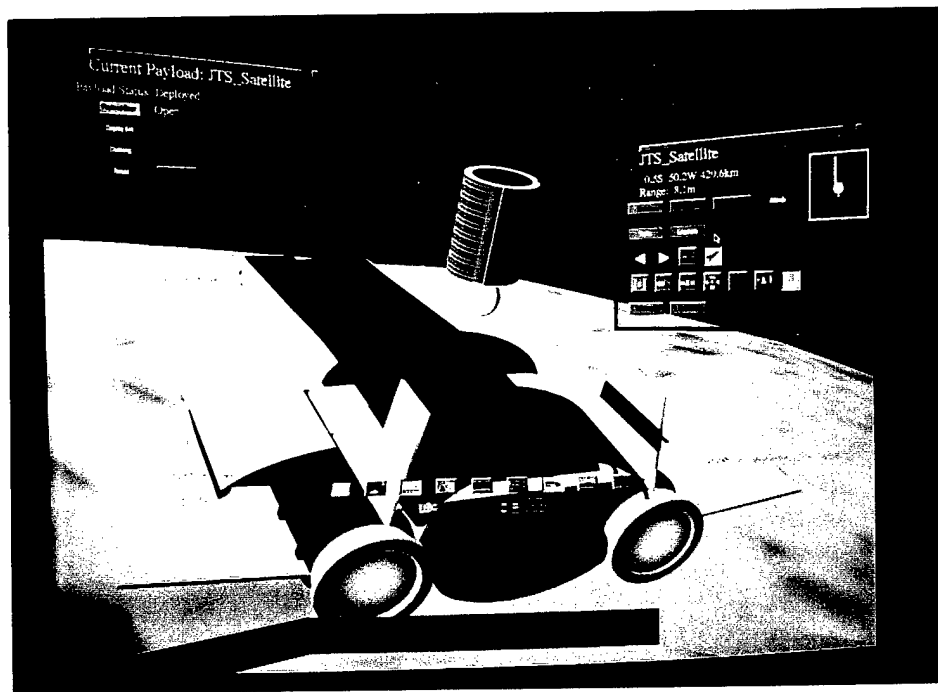
Figure 41 – Satellite after being deployed from spaceplane

User Interface

One of the primary goals of the Virtual Spaceplane research was to investigate novel ideas in cockpit design. Initially, very little was known about the style, methods, and functionality of the interface. The interface results used to fulfill each interface requirement are listed in Table 10.

Table 10 – Completion of Interface Requirements

| ID	Requirement	Resolution
Interaction Methods		
3.11	All functionality via three button mouse	Left button – geometry selection Middle button – head movement Right button – field of view
3.12	HMD with head tracking	N-Vision HMD and Ascension Bird™
Configurable Cockpit		
3.21	Selectively display information	Minimizable panels
3.22	Modify location of information	Movable panels
Displayed Information		
3.31	Gryphon state in atmosphere	Virtual HUD, Aero panel
3.32	Gryphon state during entry/reentry	Virtual HUD, Trajectory, Aero panels
3.33	Gryphon state in space	Virtual HUD, Orbit panel
3.34	State of consumables	Engineering and Payload panels
3.35	Target information	Target panel
3.36	Locating/acquiring targets	Target panel, selection of locators
3.37	System management and diagnostics	Engineering and Payload panels
3.38	Investigate hyper-text paradigms	Engineering panel
3.39	Minimize obstruction of view	Transparent panels
Controlling the Gryphon		
3.41	No throttle and stick	Mouse interaction, Virtual HUD
3.42	Change state in the atmosphere	Virtual HUD, Autopilot
3.43	Change state in space	Virtual HUD, Target panel

The hypertext interface discussed in the previous chapter enabled the completion of two of the interface requirements (3.34 and 3.38). The VSP integrates the hypertext interface into its Engineering Panel (Figure 42 through Figure 44). Through the hypertext interface, the Engineering Panel provides the status of onboard systems, consumables status, as well as checklists for use at various times through the flight. While using the hypertext interface to locate the desired information, the pilot can use the forward and back buttons to navigate through previously visited pages (similar to a conventional Internet browser). The homepage button provides a quick way to return to the main menu.

The pilot of the spaceplane must have the ability to monitor the status of mission consumables, such as fuel, air, and water. Figure 42 shows how the VSP uses the Engineering Panel to graphically represent the amount of fuel (hydrogen and oxygen) remaining onboard the spaceplane. The Engineering Panel also has similar displays for remaining air and water.
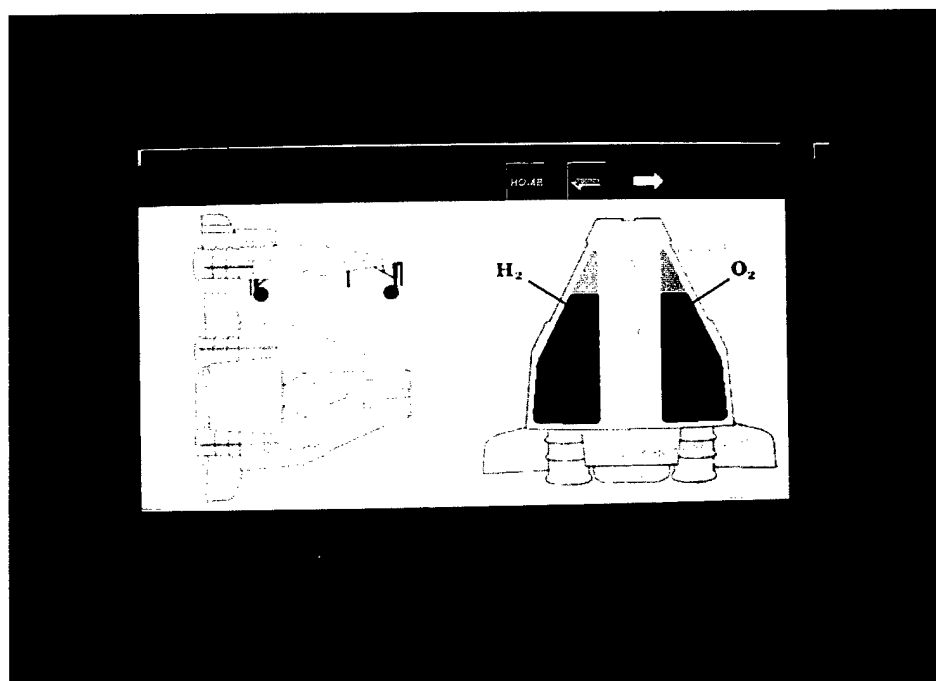
Figure 42 – Engineering Panel showing fuel consumables status

In addition to displaying consumables status, the Engineering Panel also provides an interface to access checklists useful to the pilot. The hypertext interface allows the pilot to quickly locate the desired checklist. Figure 43 shows an example of a hypertext checklist on the Engineering Panel.
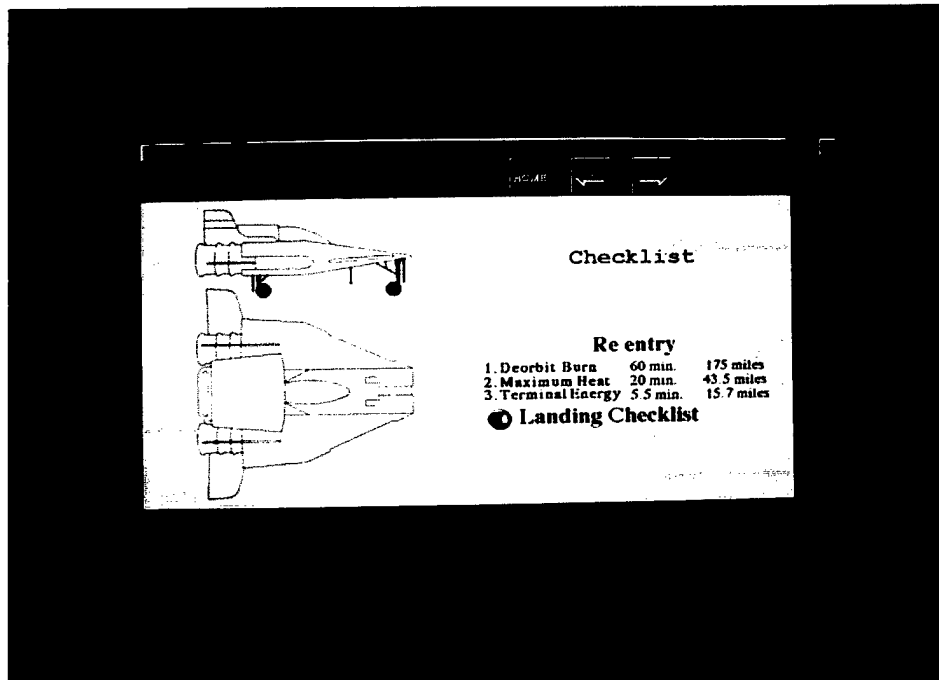
Figure 43 – Engineering Panel showing an example of a hypertext checklist

The hypertext interface on the Engineering Panel is also the primary means of obtaining the condition of the various onboard systems on the spaceplane. The left side of the display (see Figure 44) indicates a problem with the landing gear system. When the pilot selects the highlighted landing gear, additional details on the problem are displayed. The system can also provide a suggested work-around for the problem. All this information is provided in easy-to-read text, rather than warning lights or gauges as in traditional cockpits.
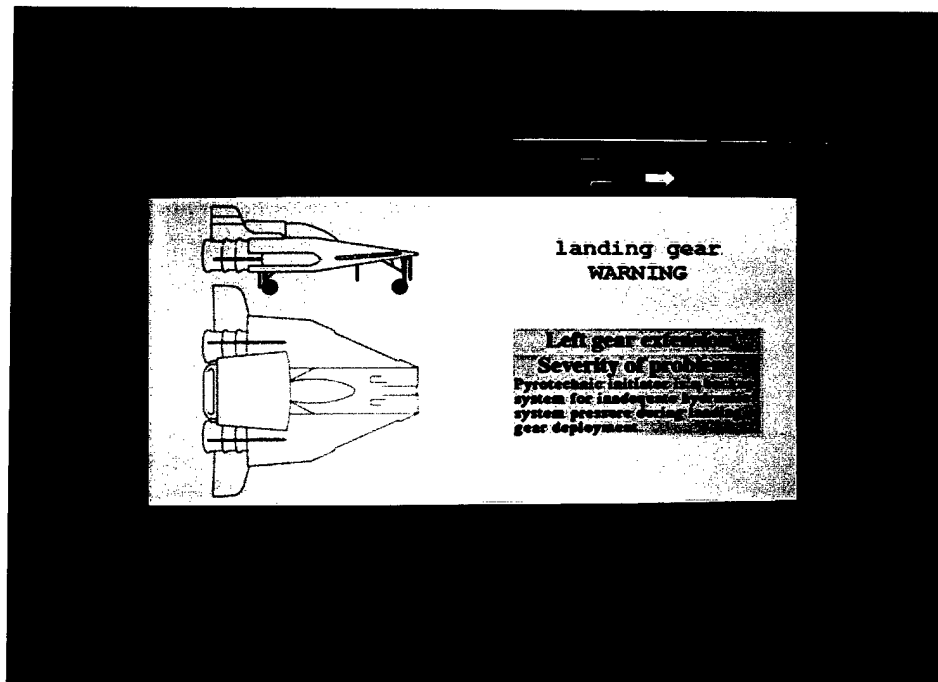
Figure 44 – Engineering Panel showing details of landing gear problem

This hypertext interface has proven to be a successful way of displaying large amounts of information in a compact region of the spaceplane's cockpit. Results on other portions of the interface design, development, and usability are given in Capt John Lewis' thesis.

Virtual Environment

The VSP developed several aspects of the environment previously unexplored at the AFIT Virtual Environments Lab. Integrating geographically accurate terrain onto a round earth resulted in the ability to immerse the user in the atmosphere, in space, or anywhere in between. The environmental requirements summary is shown in Table 11.

Table 11 – Completion of Environmental Requirements

ID	Requirement	Resolution
	Environment	
4.1	Convincing terrain near Edwards AFB	DTED based with LOD and textures
4.2	Model Earth, Sun, Moon	EARTHPROP, SUNPROP, MOONPROP
4.3	Earth orbiting objects	GPS, DMSP, DSCS, TDRS, Molniya, Space Station
4.4	Day/night, atmospheric/space transition	Blue sky, Fading Stars

The terrain in the VSP, based on actual elevation data and geographically calibrated textures, provides a realistic environment in which the spaceplane operates. The methods used to accomplish the VSP's terrain are discussed more completely in [ROTH97].

The Sun and Moon also add to the realism of the virtual environment by modeling day/night illumination differences, moon phases, and even the seasonal change in sunrise and sunset times. The correct movement of these celestial bodies in the virtual environment is accomplished using the EarthProp, SunProp, and MoonProp propagation models.

The VSP populated the VE with numerous satellite systems. Satellites from the Global Positioning System, Defense Meteorological Satellite Program, Defense Satellite Communication System, Tracking and Data Relay System, Molniya system and a space station were added to the VE. The movement of these objects is modeled using the ASTROPROP propagation model, initialized by TLEs.

Miscellaneous

Table 12 lists the miscellaneous requirements for the VSP and how the implementation satisfied them.

Table 12 – Completion of Miscellaneous Requirements

ID	Requirement	Resolution
Miscellaneous		
5.1	Accept remote entities via DIS	DISEntityManager, DISEntity, DISEntityProp, ACCU
5.2	Transmit Gryphon state via DIS	`BroadcastSimObject` in SIM, ACCU
5.3	Mean of 15 frames per second	Average of 15.6 fps

As described by Rothermel, the Virtual Spaceplane can participate in a distributed virtual environment using the Distributed Interactive Simulation (DIS) communication standards [ROTH97]. The ACCU, described in the previous chapter, performs the coordinate conversions necessary for incoming and outgoing simulation messages.

Meeting the minimum frame rate was a challenging requirement that required constant attention and optimization. The VSP's high frame rate was generally accomplished using specialized culling or levels-of-detail [ROTH97]. Also, because the VSP implements its hypertext interface using graphic images, it is important to keep the number and size of the images small to ensure a high frame rate.

Summary

The results of the research completed for the Virtual Spaceplane were covered in this chapter. Each of the requirements identified in Chapter Three have been addressed. In the next chapter, Conclusions and Future Work, we will review the work done in the VSP and consider areas where our research may be extended.

VI – Conclusions and Recommendations

This thesis presented the Virtual Spaceplane, a distributed virtual environment simulating the flight of a space vehicle from the Earth's surface to space. The Virtual Spaceplane was developed as a virtual prototype for the Military Spaceplane.

The techniques developed during this research provided the spaceplane with the ability to maneuver in the diverse operating regimes from a runway to low-earth orbit. Transitioning between the different regions was accomplished through an architecture that allowed a simulation entity to change its propagation model, along with a set of coordinate conversion routines developed during our research. The Spaceplane also proved to be an excellent testbed for analyzing new paradigms of cockpit design, including a hypertext interface used for onboard system diagnostics, checklists, and monitoring consumables.

I briefly covered some relevant background information in Chapter Two, including a description of the Military Spaceplane. In Chapter Three, the requirements of the Virtual Spaceplane were identified. The requirements were classified in five main areas: simulated capabilities of the Military Spaceplane, supported missions, plus user interface, virtual environment, and other miscellaneous requirements. Chapter Four described the research performed as well as the design and implementation of the software written to fulfill the Spaceplane's requirements. Finally, in Chapter Five I presented the results of the research and indicated how the work done satisfied the requirements identified in Chapter Three. Although the results indicate the completion of

the requirements, we identified some areas that could be improved as we worked on the project.

Recommendations for Future Work

The Virtual Spaceplane successfully simulates many of the anticipated capabilities of the Military Spaceplane. Nevertheless, several areas of the project leave room for improvement. These are discussed in the next few sections. We feel it would be useful to augment the VSP's current mission types with others projected for the Military Spaceplane. Additional enhancements could be made in the areas of the cockpit interface, enhanced autopilot capabilities, incorporating hand-tracking hardware, and the inclusion of agent technology to assist the pilot.

Incorporate Additional Mission Types

The Virtual Spaceplane currently supports two types of missions. While orbiting the Earth, the spaceplane can rendezvous and co-orbit with another object, and can deploy a satellite into low-earth orbit. When completed, the Military Spaceplane is expected to accomplish other mission types. These areas include a pop-up flight, a once-around orbit, docking with the space station, and the ability to plan a flight over a specified point on the surface of the Earth (for reconnaissance).

One type of mission trajectory that could be used by the spaceplane is referred to as a pop-up maneuver. When executing a pop-up, the VSP could deploy a payload during its exo-atmospheric trajectory, and then reenter and land downrange. The deployed payload would be boosted by an upper stage to its intended target or orbit. Adding the pop-up capability to the VSP could be accomplished by modifying the ORBITENTRYPROP

propagation model. The first portion of the mission through the lower atmosphere (using AEROPROP) would be similar to that currently performed for a multiple orbit mission. At the altitude where the spaceplane transitions from AEROPROP to ORBITENTRYPROP, the revised propagation model would route the spaceplane toward the downrange landing site rather than to orbit.

The spaceplane should also be able to launch into any azimuth and use or deploy mission assets while in a once-around orbit and return to base. This type of mission is different from a pop-up mission in that the spaceplane encircles the Earth (once) when performing a once-around mission, allowing the spaceplane to land at the same base from which it launched. Integration of the once-around orbit could be accomplished similar to the pop-up maneuver described above.

Another mission type that could be added to the Virtual Spaceplane is the capability to dock with a satellite or space station. Docking with an object in orbit involves maneuvers similar to those performed when initiating a rendezvous. However, the algorithms used for the rendezvous calculations do not currently provide the accuracy required to accomplish a docking maneuver. Providing the ability for the Virtual Spaceplane to get close enough to another object in orbit could possibly be solved by improving the accuracy of the rendezvous methods. In addition to the spaceplane and the target approaching each other slowly, the spaceplane must also be oriented correctly to ensure a safe docking procedure. Docking could be performed either manually or automatically. Manual docking would necessitate a new interface to enable the pilot to easily orient the spaceplane. Automatic docking will require modifications to the autopilot.

Finally, although the spaceplane can currently maneuver to intercept an object in space, it would also be useful to maneuver the orbiting spaceplane to pass over a specified point on the surface of Earth. This capability could be used during a reconnaissance mission (for example, to overfly Pyongyang, North Korea). Passing over a specified point over the Earth is also useful when reentering the atmosphere prior to landing. The mathematics behind a fly-over are significantly different than those used to perform a rendezvous, and would have to be developed to provide this new capability.

Modifications to the Interface

Although a significant amount of research went in to developing the spaceplane cockpit interface, this area still provides substantial areas for improvement. A few areas we identified during our research to improve the interface include the capability to perform mission planning, a mission timeline, a landing footprint display, and an interface to manipulate the waypoints used by the autopilot.

The pilot of the spaceplane currently has no guidelines for when or where to perform mission activities. This approach is in direct contrast to all spacecraft and military aircraft, which operate using a flight plan detailing the current mission. Some means of mission planning would, therefore, be a useful addition to the Virtual Spaceplane. Selection of the mission type, choosing the target, fuel requirements, and other mission planning could be accomplished prior to lift-off. Pre-flight mission planning could be achieved using flight-planning software not directly integrated into the VSP. However, in addition to pre-flight planning, the spaceplane pilot may wish to make modifications to the flight plan to account for unforeseen developments during the

mission. Allowing mid-flight retasking would necessitate adding a mission-planning panel to the spaceplane cockpit.

The Virtual Spaceplane interface could also provide a mission timeline, showing the progress along its current flight plan. The mission timeline could provide helpful information to the pilot concerning the progress of the current mission, such as time to rendezvous with the target, planned atmospheric reentry time, and periods of expected communication blackouts.

Another useful addition to the spaceplane interface would be a landing footprint display. After atmospheric reentry, the spaceplane's velocity, altitude, and remaining fuel limit the area where the spaceplane can land. The landing footprint display would allow the pilot to ensure the spaceplane can land safely at the primary destination, and would provide information useful for choosing alternate landing sites.

Allowing the pilot to adjust the route the autopilot follows could further enhance the interface. As currently implemented, the autopilot follows routes defined by waypoint that are read in from data files. The pilot could adjust the waypoints to land at a secondary landing site.

Advanced Autopilot

The Virtual Spaceplane's current autopilot is not a fully functional, intelligent autopilot. Additional research toward developing a more advanced autopilot could significantly enhance the spaceplane's effectiveness. In addition to providing the ability to manipulate waypoints (discussed in the previous section), another useful addition to the autopilot would be the capability for the autopilot to provide completely autonomous operation of the spaceplane.

A necessary precursor to autonomous operation is a fully specified flight plan. The autopilot would then be responsible for making the appropriate maneuvers at the right time during the mission. The current autopilot can automate most of the portions of the mission, but does not link them together to accomplish a completely autonomous spaceplane, from automatic takeoff, transition to orbit, orbital maneuvering, and reentry, to an automated landing.

Integrating Hand Tracking Hardware

The interface with the Virtual Spaceplane is almost entirely implemented using a head-mounted display and a mouse. Although the mouse allows for precise selection of objects in the virtual environment, the mouse is not an appropriate interface device for a cockpit. Hand tracking is perhaps a more appropriate input device in the cockpit, and could provide the Spaceplane pilot with the functionality currently provided by the mouse. Hand tracking could be integrated into the Virtual Spaceplane using the same hardware and software procedures developed for the HMD.

Agent Assistant

One of the primary research goals in the design of the VSP interface was to provide the pilot with precisely the information necessary to complete the mission, without unessential information to hamper the pilot. The Virtual Spaceplane accomplishes this capability in a limited manner using the configurable cockpit ideas. The pilot can move as well as iconify information panels as he assembles the information to accomplish tasks throughout the mission.

However, the Virtual Spaceplane may benefit by including the emerging technology of agents and associate systems. Associate systems provide decision aiding in high stress, data-heavy situations. The purpose of an associate system is not to automate more of the tasks performed on the aircraft, because automation does not necessarily help the pilot [HAMM95]. Rather, an associate system attempts to alleviate cockpit complexity, one of the high-level goals during the development of the Virtual Spaceplane.

The Virtual Spaceplane's software architecture should allow an agent assistant to be smoothly incorporated into the interface. Communication between the simulation and the agent will probably be accomplished using the Common Object Database (CODB). The agent can obtain updates and can make suggestions to the pilot using the CODB's shared memory structures.

Concluding Remarks

Research such as that done for the Virtual Spaceplane is an important step as the United States Air Force continues to pursue its expansion to include more activity in space. The Virtual Spaceplane has allowed an investigation of new ideas in cockpit design of an air-space vehicle. Furthermore, the research accomplished during the development of the Virtual Spaceplane incorporated important findings in the areas of large-scale realistic virtual environments as well as simulating the maneuvering of a vehicle in the diverse environments from earth to space. Further development of this project will assuredly bring additional insight in these areas, thereby benefiting the Air Force and the distributed virtual environment community at large.

APPENDIX A – EXTRACTING ORIENTATION INFORMATION FROM AN EULER MATRIX

Performer, a graphics programming package, provides a powerful library of vector and matrix routines. The Performer matrix mathematics can be used to multiply matrices, as well as extract information out of a matrix. Performer uses Euler angles to represent orientations. Performer provides a function, `getOrthoCoord`, which extracts the heading, pitch, and roll from an Euler matrix. This appendix details a procedure described by Thomas that allows us to obtain the same information without using Performer's math library [THOM91]. Each of the Euler angles represents rotation about one of the three primary axes. From the Euler angles, Performer can create an Euler matrix, E:

$$E = R \cdot P \cdot H \qquad (A\text{-}1)$$

where R, P, and H represent the roll, pitch and heading transform matrices respectively. The definitions of the transform matrices can be obtained from any basic graphics textbook (Equations A-2) [FOLE92][HEAR97].

$$R = \begin{bmatrix} \cos r & 0 & \sin r & 0 \\ 0 & 1 & 0 & 0 \\ -\sin r & 0 & \cos r & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (A\text{-}2a)$$

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos p & -\sin p & 0 \\ 0 & \sin p & \cos p & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (A\text{-}2b)$$

$$H = \begin{bmatrix} \cos h & -\sin h & 0 & 0 \\ \sin h & \cos h & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (A\text{-}2c)$$

In these equations, r, is the rotation about the y-axis, the pitch, p, is the rotation about the x-axis, and the heading, h, is the rotation about the z-axis. The expanded Euler matrix, E, is obtained by multiplying Equations (A-2a) through (A-2c) together:

$$E = \begin{bmatrix} E_{00} & E_{01} & E_{02} & E_{03} \\ E_{10} & E_{11} & E_{12} & E_{13} \\ E_{20} & E_{21} & E_{22} & E_{23} \\ E_{30} & E_{31} & E_{32} & E_{33} \end{bmatrix}$$

(A-3)

where

$$E_{00} = \cos r \cdot \cos h + \sin r \cdot \sin p \cdot \sin h,$$

$$E_{01} = -\cos r \cdot \sin h + \sin r \cdot \sin p \cdot \cos h,$$

$$E_{02} = \sin r \cdot \cos p,$$

$$E_{10} = \cos p \cdot \sin h,$$

$$E_{11} = \cos p \cdot \cos h,$$

$$E_{12} = -\sin p,$$

$$E_{20} = -\sin r \cdot \cos h + \cos r \cdot \sin p \cdot \sin h,$$

$$E_{21} = \sin r \cdot \sin h + \cos r \cdot \sin p \cdot \cos h,$$

$$E_{22} = \cos r \cdot \cos p,$$

$$E_{03} = E_{13} = E_{23} = E_{30} = E_{31} = E_{32} = 0, \text{ and}$$

$$E_{33} = 1.$$

It is apparent from the previous equations that the pitch, p, can be derived easily from:

$$\sin p = -E_{12}.$$

(A-4)

And, assuming $\cos p \neq 0$, the heading, h, and roll, r, are obtained from the following relationships:

$$\tan h = E_{10} / E_{11}, \text{ and}$$

(A-5a)

$$\tan r = E_{02} / E_{22}.$$

(A-5b)

If $\cos p = 0$, we assume either h or r are 0 and solve for the other value. For this example,

I'll assume $r = 0$ and solve for h using:

$$\cos h = E_{21} / E_{12}.$$

(A-6)

APPENDIX B – CONVERTING BETWEEN GEODETIC AND GEOCENTRIC COORDINATES

The ACCU routines make use of an algorithm described by Ralph Toms for converting between a geocentric (WGS84) coordinate system and a geodetic (latitude/longitude/altitude) system [TOMS93]. The conversion from geodetic to geocentric coordinates is straightforward. The inverse transformation is performed using an efficient iterative algorithm.

Position in our geocentric coordinate system is defined by the vector (X, Y, Z) where X, Y, and Z are in units of meters. The geodetic coordinate system defines position using the vector (ϕ, λ, h) where ϕ and λ represent the latitude and longitude measured in radians, and h is the altitude above the surface of the Earth.

The Earth is represented as an oblate spheroid with the major axis, **a**, (at the equator) is 6,378,137.0 meters and the minor axis, **c**, (at the poles) is 6,356,752.3142 meters. The radius of the Earth at any point, R_N, is therefore a function of latitude, as shown in Equation (B-1).

$$R_N = \frac{a}{\sqrt{1 - \varepsilon^2 \sin^2 \phi}}, \qquad (\text{B-1})$$

where the eccentricity, ε, is defined by

$$\varepsilon^2 = \frac{a^2 - c^2}{a^2}.$$

Given the geodetic coordinates (ϕ, λ, h), the geocentric coordinates can now be calculated quite easily using Equations (B-2) through (B-4).

$$X = (R_N + h)\cos \phi \cos \lambda \qquad (\text{B-2})$$

$$Y = (R_N + h)\cos \phi \sin \lambda \qquad (\text{B-3})$$

$$Z = \left(R_N \frac{c^e}{a^2} + h \right) \sin \phi \qquad\qquad \text{(B-4)}$$

However, converting from a geocentric position, given by the vector (X, Y, Z), to geodetic requires an iterative approach. Toms' approach uses the Bowring method, which provides quick, accurate results. First, the geodetic latitude, λ, is computed directly using Equation (B-5).

$$\lambda = \tan^{-1}\left(\frac{Y}{X} \right)$$
$$\left(-\frac{\pi}{2} \le \lambda \le \frac{\pi}{2} \right) \qquad\qquad \text{(B-5)}$$

It now becomes useful to define some intermediate values.

$$W = \sqrt{X^2 + Y^2}$$

$$f = \frac{a - c}{a}$$

$$\varepsilon'^2 = \frac{a^2 - c^2}{c^2}$$

The latitude is now solved iteratively using the Bowring method by introducing an auxiliary variable β such that

$$\tan \phi_{i+1} = \frac{Z + c \cdot \varepsilon'^2 \cdot \sin^3 \beta_i}{W - a \cdot \varepsilon^2 \cdot \cos^3 \beta_i} \qquad\qquad \text{(B-6)}$$

$$\tan \beta_{i+1} = (1 - f) \tan \phi_{i+1} \qquad\qquad \text{(B-7)}$$

with the initial value of β given by

$$\tan \beta_0 = \frac{a \cdot Z}{c \cdot W}. \qquad\qquad \text{(B-8)}$$

94

The iteration is terminated when $\left| \tan \phi_{i+1} - \tan \phi_i \right|$ is small enough and the latitude, ϕ, is then computed by using the inverse tangent function. Given the latitude, the height can be computed using Equations (B-1) and (B-9).

$$h = \frac{W}{\cos \phi} - R_N \qquad \text{(B-9)}$$

APPENDIX C – DETAILED VIRTUAL SPACEPLANE REQUIREMENTS

The following tables were used during the development of the Virtual Spaceplane. Because the project was developed using rapid prototyping techniques, the detailed requirements which follow were continuously being modified with items being added and removed from the list as research showed particular avenues to be more productive than others. This appendix should prove useful to anyone pursuing future research on the Virtual Spaceplane.

VSP CAPABILITIES

1	**Emulate Mark III version of MSP**
	✓ Take-off/Landing runway minimum size: 8000ftx150ft
	✓ Ferry Capability
	✓ Orbit Entry
	❖ Once-around Orbit
	✓ Full six-DOF translation and rotation in orbit
	✓ Excess On-board propellant (ΔV) of 600fps.
	✓ Pointing accuracy of 10 milliradians.
	✓ Rendezvous, co-orbit capable
	❖ Docking capable.
	✓ Mission duration of 24-72 hours
	✓ Payload bay capable of housing standardized payload container (25' x 12' x 12'), weight 40 klbs
	✓ NAVSTAR/GPS navigation aids.
	✓ Automated Test and Checkout of subsystems and payload
	✓ Spaceplane able to be directed from onboard or from the ground
	❖ Capable of Autonomous control
1	**Horizontal Takeoff/Landing**

Legend:

	Priority
1	Vital
2	Important
3	Lesser

	State
	Complete
□	Incomplete
□	In Progress
	Future Work

	Accurate Flight Characteristics
2	**Runway Operations** ✓ Taxi to/from runway ✓ Terrain following (can operate on non-level runways) ✓ Provides realistic acceleration, braking characteristics
2	**Air Operations** ❖ Models Spaceplane aerodynamic characteristics ✓ Allows access to high altitude and speed necessary for orbit entry
1	**Orbit Entry**
1	**Space Operations** ✓ Accurate astrodynamic model
1	**Orbital Changes** ✓ VSP can change orbital parameters
1	**Reentry**
3	**Sub-orbital flight (Exo-atmospheric)**
1	**Ability to switch between flight models** ✓ Transition from runway to air ✓ Transition from air to space ✓ Transition from space to air ✓ Transition from air to runway
	Manual Flight
3	**Manually fly through atmosphere**
1	**Manually specify orbit modifications**
3	**Manually dock with Space Station**
	Automatic Flight
2	**Auto-Takeoff**
2	**Auto-AeroFlight** ✓ Ability to view waypoints
2	**Auto-Orbit**
3	**Auto-Dock**
2	**Auto-Land**
2	**Switch between manual & automatic flight**
2	**Autonomous Command** ❖ Complete autopilot. User pushes button and sits back.
1	**Day/Night Operations**
1	**Single Pilot**
3	**Simulated Communications (no voice capabilities)** ❖ Using a "text box" approach
3	**Simulated uplink of Mission Retasking**

3	**GPS Navigation** ✓ GPS Satellites in orbit ✓ GPS Position and Velocity displayed to pilot
2	**DIS/HLA Capable** ✓ Receive DIS/HLA data packets ✓ Display DIS/HLA entities in environment ✓ Transmit position of VSP via DIS

VSP MISSION PROFILES

2	**Mission Planning Panel** ❖ Used at startup of Gryphon ❖ Choose from list of specified mission profiles ❖ Select from "Start on runway" or "Start in Orbit" ❖ Perhaps using stand-alone program such as UCPOP
1	**Once Around** ❖ Takeoff and Land at Edwards AFB
3	**Multiple Orbit** ✓ Takeoff and Land at Edwards AFB
	Rendezvous
3	***Dock with Space Station*** ✓ Provide visual of Space Station ❖ Details on docking progress
2	***Co-orbit with satellite*** ✓ Provide visual of Satellite
	Payloads ✓ Payload doors open/close ✓ Provide visual of payload ✓ Payload status details
1	***Deploy Satellite***

VSP INTERFACE

	Immersive Virtual Environment User Interface
1	***Ability to control interface with/without keyboard/mouse***
1	***Use of Head Tracking hardware***
1	**Use of Head Mounted Display** ✓ Gauges and displays can be read with/without HMD
2	***Use of Finger Tracking hardware*** ❖ "Velcro Finger" to select and move objects in environment ❖ All controls operable with engloved hands

	Configurable Cockpit ✓ User presented with semi-transparent console signifying front of Spaceplane ✓ Panels generally placed on or near consoles. ✓ Ability to control/monitor Spaceplane from cockpit.
2	***Panel placement*** ❖ Anywhere on Virtual HUD (Velcro Finger) ✓ Stuck on console (Mouse)
2	***Minimize Panels*** ✓ Panels can be minimized using button on panel ✓ Panels can be turned on and off using toolbar buttons
	Graphic Displays
2	***Takeoff/Landing Profile – Plan View*** ✓ Shows position of Gryphon at airfield
2	***Landing Footprint Display***
1	***Orbit Profile (Mercator display)*** ✓ Ground Trace Trails for previous position of Gryphon ✓ Projected ground trace based on current orbital parameters
1	***Orbital Elements Display*** ✓ 3D Graphic Panel ✓ Trajectory and current position/orientation of Gryphon ✓ Trajectory and current position of Target ✓ Ability to rotate display for more convenient display of orbit ✓ Display is resizable ✓ Separate orbit color for target vehicle
2	***Vertical Velocity Indicator*** ✓ Indication of the magnitude of our vertical velocity

1	***Attitude/Direction Indicator (ADI)***
	✓ Similar to Wright Labs'
	✓ Display of Heading/Pitch/Roll
	     Plane symbol remains fixed, horizon moves to show pitch/roll
	     Arrow around ADI circle denotes Heading
	✓ Display of Speed/Altitude
	     Grid moving across ADI Ground signifies ground speed
	     Higher altitude represented by grid lines getting closer together

2	***Consumables Status*** ✓ Fuel (Delta-V remaining) ✓ Oxygen (time remaining) ✓ Water

	Virtual HUD (Heads Up Display)
	✓ Provides Alternative to a window ✓ Virtual environment displayed through virtual window ✓ Additional HUD information
1	**Manual Flight** ✓ Provides capability to maneuver Spaceplane manually in atmosphere
1	**Space Control** ✓ Display of potential transfer orbits (Δv and time for each) ✓ Display of intercept point ✓ Display of orbit intersection point
3	**Manual Docking** ❖ Provides capability to manually dock with Space Station
2	**Target Symbols** ✓ Symbols denoting affiliation (friendly/enemy/unknown) of entities ✓ Perhaps also denoting proximity
2	**Waypoint Symbols** ✓ Graphical display of waypoint symbols "out the window" for proposed route ✓ Ability to view waypoints on Takeoff/Landing Plan View and Orbit Profile ❖ Ability to move existing waypoints ❖ Ability to add new waypoints
2	**Course Guides** ❖ ("Fly through the squares") ✓ Final Approach and Landing Corridor Frustum Changes color to indicate accuracy of approach ❖ Approach Cylinder depiction
2	**Click on objects to get info** ✓ Click on satellites, space station ✓ Provides Name, Image, Range, Time to intercept, Bearing, Elevation Angle
3	**Aerodrome Indicator** ✓ Graphical representation of Edwards AFB aerodrome Helps user see field from distance
3	**Display of Sensor Fan**
3	**Display of Satellite coverage**

	HTML Interface ✓ Pseudo-HTML Hypertext interface ✓ Home page, backward, forward control ✓ Provides on-line help
1	***Onboard Systems display (Engineering Panel)*** ✓ Status of onboard systems (Red/Green) ✓ Provides details on system complications ✓ Provides checklist of suggested repair actions
1	**Checklists** ✓ Preflight checklist ✓ Repair checklist ✓ Perhaps implemented using HTML interface
2	**Timeline** ✓ Clock ❖ Take Off ❖ Way Points ❖ Entry into Orbit ❖ Reentry ❖ Landing ❖ Light/Dark (in sunlight/behind Earth) ❖ Mission Specific ❖ Weapon Launch ❖ Payload Deployment ❖ Rendezvous
2	**Trails on objects** ✓ Trails show previous position of object ✓ Trail color represents affiliation of object (friend/foe) ✓ User has ability to turn trails on/off
2	**Locators on objects** ✓ Symbol representing type/affiliation/proximity of object ✓ User has ability to turn locators on/off.
3	**Agent assistant** ❖ Makes suggestions to pilot ❖ Capable of minimizing panels ❖ Display high-priority panels when emergency conditions occur.

VSP ENVIRONMENT

1	**Graphical Display of Terrain using multiple LOD** ✓ Multiple Levels of Detail ✓ High Quality Textures
2	**Realistic representation of Edwards AFB airfield** ✓ Runways ✓ Control Tower ✓ Taxiways
	Satellites displayed in correct orbits ✓ 3D Model of satellites ✓ Using accurate astrodynamic code ✓ Reads in NASA/NORAD two-line element files
1	*GPS*
2	*Others* ✓ DMSP ✓ DSCS III ✓ TDRS ✓ Molniya
2	**Space Station in correct orbit** ✓ 3D Graphical model of Space Station ✓ Use International Space Station in proposed orbit
3	**Nearby Planets in correct orbits**
1	**Sun & Moon in correct orbits**
1	**Stars and Constellations** ✓ Over 32000 stars in correct location ✓ Stars displayed with varying levels of magnitude and color ✓ Over 80 constellations displayed in night sky with constellation names
2	**Display of Day/Night terminator**
2	**Pseudo-Atmosphere** ✓ Sky transitions from blue to black with sunset ✓ Sky transitions from blue to black with altitude

VSP USABILITY REQUIREMENTS

1	**Interface Design Specifications** ✓ Specify details of interface
1	**Interface Design Guidelines** ✓ Specify how displays are constructed

VSP HARDWARE/SOFTWARE REQUIREMENTS
Requires Silicon Graphics Onyx or Onyx II
- ✓ Reality Engine/ Infinite Reality
- ✓ Best results with 4 processor 250MHz R4400 Reality Engine with 16Mbyte Texture Memory
- ✓ OpenGL/Iris GL capable
- ✓ Requires Performer 2.0 Execution Environment

Frame Rate
- ✓ At least 15fps 90% of the time

VSP LIMITATIONS
Mission Profiles
- Takeoff/Landing from Edwards AFB only
- Limited Target Locations
- No ability to retrieve satellites
- No Pop-up Missions
    Popups require takeoff and landing at different locations.
- No Auto-Ferry Capability
- No Mission Abort options
    Will provide "PANIC" button to jump back to earth.

Capabilities
- No Vertical Takeoff/Landing Capability
- No Egress Capabilities
- No Threats

Interface
- No Capability for co-pilot seat
- No HOTAS
- No attempt to integrate video
- HTML Interface has no capability to use Internet.

Environment
- No weather near Earth's surface

BIBLIOGRAPHY

[ADAM96] Adams, Terry A. *Requirements, Design, and Development of a Rapidly Reconfigurable, Photo-Realistic, Virtual Cockpit Prototype.* MS Thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, AFIT/GCS/ENG/96D-02, December 1996.

[AFA97] "US Air Force Almanac 1997". *Air Force Magazine*, Air Force Association, May 1997.

[BATE71] Bate, Roger R., Donald D. Mueller, and Jerry E. White, *Fundamentals of Astrodynamics*, New York, Dover Publications, 1971.

[BLOO74] Bloom, Richard L. *An Algorithm for Minimum-Fuel Two-Impulse Rendezvous.* MS Thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, AFIT/GA/MC/74-2, September 1974.

[BRYS93] Bryson, Steve. "Introduction". *Implementing Virtual Reality*, Course Notes 43. SIGGRAPH Conference on Computer Graphics and Interactive Techniques, Anaheim, CA, 1993.

[DOD87] Department of Defense, *World Geodetic System 1984 (WGS84), Its Definition and Relationships with Local Geodetic Systems.* DMA TR 8350.2, Washington: Defense Mapping Agency, 1987.

[ERIC93] Erichsen, Matthew Nick. *Weapon System Sensor Integration for a DIS-Compatible Virtual Cockpit.* MS Thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, AFIT/GCS/ENG/93-07, December 1993.

[FOGE96] Fogelman, Ronald R., Chief of Staff of the United States Air Force. "Global Engagement." Speech presented at the Smithsonian Institution, Washington DC. Available via the Internet. www.af.mil/news/Nov1996/n19961122_961185.html, 21 November 1996.

[FOLE92] Foley, James D., Andries van Dam, Steven K. Feiner, John F. Hughes. *Computer Graphics: Principles and Practice*, Second Edition. Reading MA, Addison-Wessley Publishing Company, November 1992.

[GARC96] Garcia, B., *Design and Prototype of the AFIT Virtual Emergency Room: A Distributed Virtual Environment for Emergency Medical Simulation*, MS Thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, AFIT/GCS/ENG/96D-07, December 1996.

[HAMM95] Hammer, John M., and Ronald L. Small, "An Intelligent Interface in an Associate System", *Human/Technology Interaction in Complex Systems*, Volume 7, 1995. pp. 1-44.

[HEAR97] Hearn, Donald, M. Pauline Baker. *Computer Graphics, C Version.* Upper Sadie River, NJ, Prentice Hall, 1997.

[HOOT80] Hoots, Felix R. and Ronald L. Roehrich, "Models for Propagation of NORAD Element Sets", *Space Track Report No. 3*, Peterson AFB, Aerospace Defense Command. Available via the Internet. www.grove.net/~tkelso/NORAD/documentation/spacetrk.pdf, December 1980.

[HUTS97] Hutson, Larry, *A Representation Approach to Knowledge and Multiple Skill Levels for Broad Classes of Computer-Generated Forces*, MS Thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, AFIT/GCS/ENG/97D-09, December 1997.

[IST93] *Proposed IEEE Standard Draft Standard for Information Technology— Protocols for Distributed Interactive Simulation Applications Version 2.0 Second Draft*. Orlando FL, Institute for Simulation and Training, March 1993.

[KERR88] Kerry, Mark Joels, Gregory P. Kennedy. *The Space Shuttle Operator's Manual*. New York NY, Ballantine Books, 1988.

[KUPE92] Kuperman, Gilbert G., *Information Requirements Analyses for Transatmospheric Vehicles*, Defense Technical Information Center, June 1992.

[LANE79] Lane, M.H. and F.R. Hoots, "General Perturbations Theories Derived from the 1965 Lane Drag Theory", *Project Space Track Report No. 2*, Peterson AFB, Aerospace Defense Command, December 1979.

[LEWI97] Lewis, John M. Requirements, *Design and Prototype of a Virtual User Interface for the AFIT Virtual Spaceplane*, MS Thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, AFIT/GM/ENG/97D-02, December 1997.

[LOPE93] Loper, M. and S. Seidensticker, *The DIS Vision: A Map to the Future of Distributed Simulation*. Technical report, Institute for Simulation and Training, Orlando FL, 1993.

[MCKI91] McKinnon, G.M., and R. Kruk. "Multiaxis Control of Telemanipulators," *Pictorial Communication in Virtual and Real Environments*, Taylor and Francis, London, 1991.

[MEEU91] Meeus, Jean, *Astronomical Algorithms*. Richmond VA, Willmann-Bell, 1991.

[MSIC97] Military Spaceplane Integrated Concept Team, Science & Technology Panel, *Technology Roadmap for a Military Spaceplane System*, Draft version 1.1, 3 April 1997.

[NASA94] *Shuttle Crew Operations Manual*. Version 1.5. National Aeronautics and Space Administration, July 1994.

[PAUS97] Pausch, Randy, Dennis Proffitt, and George Williams, *Quantifying Immersion in Virtual Reality*, Computer Graphics Proceedings, Annual Conference Series, 1997.

[ROLF86] Rolfe, J.M., K.J. Staples, *Flight Simulation*, Cambridge, Great Britain, Cambridge Univ. Press, 1986.

[ROTH97] Rothermel, Scott A. *Architecture, Design and Implementation of a Rapidly Prototyped Virtual Environment for a Military Spaceplane*, MS Thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, AFIT/GCS/ENG/97D-17, December 1997.

[RUMB91] Rumbaugh, James, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen, *Object-Oriented Modeling and Design*, Englewood Cliffs, New Jersey, Prentice-Hall, 1991.

[SGI95] *IRIS Performer Programmer's Guide*. Silicon Graphics, Inc, 1995.

[SSA97] Soaring Society of America, acro.harvard.edu/SSA/BGA/os_note.html, 1997.

[STYT96] Stytz, Martin R, Elizabeth G. Block, Brian B. Soltz, and Kirk Wilson, "The Synthetic Battle Bridge: A Tool for Large-Scale VEs," *IEEE Computer Graphics and Applications*, January 1996. pp. 16-26.

[STYT97] Stytz, Martin R, Terry Adams, Brian Garcia, Steven Sheasby, and Brian Zurita, "Rapid Prototyping for Distributed Virtual Environments," *IEEE Software*, September/October 1997. pp. 83-92.

[SWEE97] Sweetman, Bill, "Spies in the Sky," *Popular Science*, April 1997. pp. 42-48.

[THOM91] Thomas, Spencer W., "Decomposing a Matrix into Simple Transformations," *Graphics Gems II*, Edited by James Arvo. San Diego CA, Academic Press, 1991. pp. 320-323.

[TOMS93] Toms, Ralph M., "An Efficient Algorithm for Geocentric to Geodetic Coordinate Conversion," *Proceedings on Standards for the Interoperability of Distributed Simulations, Volume II*, (13th Proceedings). Orlando FL, Institute for Simulation and Training, September 1995. pp. 635-642.

[USAF92] *Air Force Manual 1-1, Volume II*, United States Air Force, March 1992.

[USC90] U.S. Congress, Office of Technology Assessment, *Access to Space: The Future of U.S. Space Transportation Systems*, OTA-ISC-415, Washington, DC, U.S. Government Printing Office, April 1990.

[VERD97] Verderame, Ken Maj, Maj Andrew Dobrot, *System Requirements for a Military Spaceplane*, Phillips Laboratory Space Technology Directorate, DRAFT version 1.0, 24 April 1997.

[WELL96] Wells, William D., *Collaborative Workspaces within Distributed Virtual Environments*, MS Thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, AFIT/GCS/ENG/96D-26, December 1996.

[WILL96] Williams, G., *Solar System Modeler: A Distributed, Virtual Environment for Space Visualization and GPS Navigation*, MS Thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, AFIT/GCS/ENG/96D-29, December 1996.

[ZELT92] Zeltzer, David, "Autonomy, Interaction, and Presence," *Presence: Teleoperators and Virtual Environments*, Volume 1, Number 1, 1992. pp. 127-132.

[ZURI96] Zurita, Vincent Brian. *A Software Architecture for Computer Generated Forces in Complex Distributed Virtual Environments*. MS Thesis, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, AFIT/GCS/ENG/96D-32, December 1996.

VITA

Lt Troy D. Johnson was born on 7 August 1963 in St. Louis Park, Minnesota. He graduated from Glencoe Senior High School in 1983 and enlisted in the Air Force on 17 October 1984. While enlisted, he was assigned to Bitburg AB, Germany and Brooks AFB, Texas. In 1991 he was selected for the Airmen's Education and Commissioning Program (AECP), and was reassigned to Texas A&M University to complete his undergraduate requirements. He graduated Summa cum Laude with Bachelor of Science degrees in Meteorology and Computer Engineering in May 1994. He received his commission on 30 September 1994 upon graduation from Officer Training School.

He then served as a System Manager for TACC Weather Directorate at Scott AFB, IL. In June 1996, he entered the Graduate School of Engineering, Air Force Institute of Technology. He will graduate with a Master of Science degree in Meteorology with an emphasis in graphical design in December 1997. He will then be reassigned to the Air Force Global Weather Center, Offutt AFB, Nebraska.

Permanent Address: 210 6th St N.
                       Hudson WI 54016

Permanent email: troy.johnson@acm.org

REPORT DOCUMENTATION PAGE			

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE December 1997	3. REPORT TYPE AND DATES COVERED Master's Thesis	

4. TITLE AND SUBTITLE
The Virtual Spaceplane: Integrating Multiple Motion Models and Hypertext in a Virtual Environment

5. FUNDING NUMBERS

6. AUTHOR(S)
Troy D. Johnson, First Lieutenant, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
2750 P Street
WPAFB, OH 45433-7126

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GM/ENG/97D-01

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
PL/VTS
Mr Jerry Gibson
3550 Aberdeen Ave SE
Kirtland AFB, NM 87117
email:; JERRY_GIBSON%PL-07M2@ccmail.plk.af.mil

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The Air Force is currently investigating the possibility of developing a manned vehicle capable of operating in space. This Military Spaceplane (MSP) will be capable of ascent to low-earth orbit and maneuvering while in orbit. The goal of this research involved creating the Virtual Spaceplane (VSP), a virtual environment (VE) simulator for the MSP. This thesis examines two ideas significant to virtual environments and cockpit design: multiple motion models and hypertext in a VE.

Movement in a VE has traditionally been modeled using a single motion model. Little work has been done to allow a change of the motion model used during the simulation. This thesis suggests partitioning simulation entities into two sections: the geometry model and the propagation model. This approach is demonstrated in the VSP using multiple propagation models as it transitions from runway to orbit.

This thesis also examines the use of hypertext within a VE. Hypertext has been shown useful for readers to quickly locate information. This thesis will discuss the integration of a hypertext interface into the VSP. The hypertext interface provides checklists, systems status, and consumables status. Hypertext provides the spaceplane pilot with an effective means of referencing large amounts of data.

14. SUBJECT TERMS
Modeling, Simulation, Virtual Environments, Space Flight, Spaceplane, Hypertext

15. NUMBER OF PAGES
121

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL